Dyna gen .EXE (This program links with DynaLib.dll)

1

Attachment

```
/***************************************************************
  main.c  (Dynagen function for main)
        By Takashi Kosaka (C) SegaSoft INC.    1997 -

  SEGASOFT CONFIDENTIAL - Unpublished Copyright (c) [1997],
  SegaSoft, Inc   All Rights Reserved.
***************************************************************/

#include <stdio h>
#define WINDOWS
#ifdef WINDOWS
#include <windows.h>
#endif
#include "cfs.h"

// Function Define
void init_scheme();
int CFSMountWithPath(char *path);
  t CreateCFS(char *path);
   igned long DynaEvalString(char *eval_form, int *type);
  ar * CreateReletivePath(char *src_path, char *dst_path);
void MakeAppPath(char *DefFilePath,char *AppName,char *AppPath);
void GetRealAPPName(char *appname);
void ReadMultiFile(char *app_def_path),
void WriteMulti();
void UpdateMultiFile(char *DllPath,char *AppName, char *UpdateList);
char *MakeDquteString(char *string);
void StreamPrintf(char *format,char *arg1, char *arg2);
char * StreamPrintEnd();
void CreateShipFile(char *AppDefPath,char *AppName,char *DllPath,char *DllName,char *UpdateList);


/* Line management Table in  def file */
typedef struct _Line {
    char *fname;
    char *no;
    int option;
    char *add;
    struct _Line *next;
} Line, *PLine;


/* Find source string from target string */
int SearchString(char * source,char *target)
{
  int i,j,slen,tlen,
  slen = strlen(source);
  tlen = strlen(target);
  j = 0;
  for(i = 0 ; i < tlen ; i++ ) {
    if(*(target + i) == *(source + j)) {
      j++,
      if(j >= slen)
        return(1),
    }
    else
      j = 0;
  }
  return(0);
}

PLine SearchLine(char *func,PLine top)
{
  PLine now,
  for(now = top ; now , now = now->next) {
    if(strcmp(func,now->fname) == 0)
      return((PLine)now),
  }
  return((PLine)NULL),
}

PLine SearchStringLine(char *func,PLine top)
{
  PLine now,
  for(now = top ; now , now = now->next) {
    if(SearchString(func,now->fname))
```

```
      return((PLine)now);
    }
  return((PLine)NULL);
}

/* Read One Line */
int ReadLine(FILE *fp,char buf[],int max_size)
{
  int i,data,
  for(i = 0 ; i < max_size , ) {
    data = getc(fp),
    if(data == FOF) {
      return(EOF);
    }
    else if(data == '\n') {
      if(i == 0) {
        continue;
      }
      return(i);
    }

    buf[i++] = data;
    buf[i] = '\0';
  }
 // buf[i - 1] = '\0';
  return(i),
}

char *Stralloc(char *buf)
{
  char *new_buf;
  if((new_buf = (char *)malloc(strlen(buf) + 1)) == NULL) {
    return((char *)NULL);
  }
  strcpy(new_buf,buf),
  return((char *)new_buf);
}

/* Set correct data into Line Data. This is sequentail setting */
void SetIntoLine(char *src,PLine line)
{
  if('line->fname)
    line->fname = Stralloc(src);
  else if ('line->no) {
    line->no = Stralloc(src + 1);
  }else if('line->option)
    line->option = 1,
  else if('line->add) {
    if(*src == '0')
      *src = '#';
    line->add = Stralloc(src);
  }
}

/* Create Line Data from String */
PLine CreateLineData(char *data)
{
  int i,len,code,
  char *start,

  PLine now,
  if((now = (PLine)malloc(sizeof(Line))) == NULL) {
    printf("Can not allocate Line memory !!!\n");
    exit(-1),
  }
  now->fname = (char *)NULL;
  now->no     = (char *)NULL,
  now->option = 0;
  now->add    = (char *)NULL,
  now->next   = (PLine)NULL,
  code = 0,
  len = strlen(data),
  start = data;

  for(i = 0 , i < len + 1 , i++ ) {
    if((*(data + i) != ' ' || *(data + i) == '\0') && code == 0) {
      code = 1;
      start = data + i;
    }
    if(code == 1 && (*(data + i) == ' ' || *(data + i) == '\0')) {
      code = 0;
      *(data + i) = '\0';
      SetIntoLine(start,now),
    }
  }
  return((PLine)now);
}

/* Load def into memory */
PLine  LoadDefFileIntoMemory(FILE *fp)
{
  PLine top,now,old;
  char buf[2048];
  int err;

   ld = top = (PLine)NULL;
  err = ReadLine(fp,buf,2048),
  while (err != EOF) {
    if(buf[0] == ';')
      break;
    now = CreateLineData(buf);
    if(old) old->next = now;
    if(!top) top = now;
    old = now;
    err = ReadLine(fp,buf,2048);
  }
  return((PLine)top);
}


void vprint(CF *fp,char *cont,char *arg1,char *arg2)
{
  char buf[2048];
  int len,i,

  for(i = 0 ; i < 2048 ; i++ )
      buf[i] = '\0';

  if(arg1)
    sprintf(buf,cont,arg1,arg2);
  else
    sprintf(buf,cont);

  len = strlen(buf),
  cfs_encode_write(buf,len,1,fp);
}

// Clear Buffer put '\0'
 'd clear_buf(char *buf,int size)

    int i,
    for(i = 0 ; i < size , i++ ) {
        *(buf + i) = '\0';
    }
}

// Global
PLine App,

char *FindFunctionNameFromNumber(char *number)
{
    PLine now,
    for(now = App ; now ; now = now->next) {
        if(strcmp(now->no,number) == 0) {
            return((char *)now->fname);
        }
    }
    return((char *)NULL);
}


/***************************************************************
```

```
DynaGen: Create the virtual file system and script in
the virtual file system.
Argments.  dynagen application-path-name <dynaobj-path-name>
If sencond argment is given, dynagen generates script file
in the virtual file system
application-path-name : relative path name for the existing application
dynaobj-path-name     : relative path name for the existing dynamodule
***************************************************************/
void main(int argc , char *argv[])
{
  PLine app,dll,now,target;
  char appname[1024];
  char dllname[1024];
  char appath[1024];
  char dllpath[1024];
  char temp[2048];
  char *DllReletivePath,*UpdateList;
  FILE *fp;
  CF   *out_v;
  int ret_v;
  int type;
  int need_script;

  need_script = 1;

  if(argc < 2 || argc > 3) {
    printf("generator: application-DEF-file-path dynamodule-DEF-file-path or\n");
    printf("generator: application-DEF-file-path.\n");
    exit(-1);
  }

  if(argc == 2)
    need_script = 0; /* Do not need to make a script file */

  if((fp = fopen(argv[1],"r")) == NULL) {
    printf("generator: Can not find %s\n",argv[1]);
    exit(-1);
  }

  ReadLine(fp,temp,1024),    /* Full path of an App DEF File */

  ReadLine(fp,appname,1024);  /* Get App Name */
  GetRealAPPName(appname);

  MakeAppPath(temp + 1,appname,appath); // Create AppPath

  ReadLine(fp,temp,1024);       /* Exports string ignore */
  App = app = LoadDefFileIntoMemory(fp);
  fclose(fp);

  if(need_script) { /* Need to make the script file into VFS */
    if((fp = fopen(argv[2],"r")) == NULL) {
      printf("generator: Can not find %s\n",argv[2]),
      exit(-1);
    }
    ReadLine(fp,temp,1024), /* Full path of a DLL DEF file */
    ReadLine(fp,dllname,1024),  /* Get DLL Name */
    GetRealAPPName(dllname);
    // Create DLL Path
    MakeAppPath(temp + 1,dllname,dllpath),

    ReadLine(fp,temp,1024),       /* Exports string ignore */
    dll = LoadDefFileIntoMemory(fp);
    fclose(fp);

    DllReletivePath = CreateReletivePath(appath,dllpath);

    ret_v = CFSMountWithPath(appath);
    ReadMultiFile(argv[1]); // DynMulti data
    switch(ret_v) {
       case 0  /* Mount Success */
          if((out_v = cfs_open("init scm",CO_WRONLY)) == NULL) {
               printf("Something Wrong Virtual File system \n"),
               printf("Can not make file in the VFS\n"),
               exit(-1),
          }
```

```
            break;
        case 1· /* Wrong Path Name Use */
            printf("Wrong path Name use in %s \n",appath);
            exit(-1);
        case 2  /* Different VFS use */
            printf("Wrong path Name use in %s \n",appath);
            exit(-1);

        case 3. /* Does not exits VFS */
            if('CreateCFS(appath)) {
                if((out_v = cfs_open("init.scm",CO_WRONLY)) == NULL) {
                    printf("Something Wrong Virtual File system  \n");
                    printf("Can not make file in the VFS %s\n",appath);
                    exit(-1);
                }
            }
            else {
                printf("Access denied in %s \n",appath);
                exit(-1);
            }
            break,
    }

//  DllReletivePath = CreateReletivePath(appath,dllpath);
    /* SCM file */
    vprint(out_v,"(enable-dynamod \"%s\" \"%s\" '(\n",
        DllReletivePath,appname);

    init_scheme(); // Initialize Scheme


    StreamPrintf("'(",NULL,NULL),
    // Swapping Function Set
    for(now = dll ; now ; now = now->next ) {
      if((target = SearchLine(now->fname,app)) != NULL) {
        vprint(out_v,"          (%s . %s) \n",target->no,now->add);
        StreamPrintf("(%s . %s) \\\n",target->no,now->add);
      }
    }

    if(target = SearchStringLine("DeleteDynaMod",dll)) {
        vprint(out_v,"            (%s . %s) \n","-1",target->add);
        StreamPrintf("(%s . %s) ","-1",target->add);
    }
    StreamPrintf(")",NULL,NULL);
    vprint(out_v,"   ))\n",NULL,NULL);
    cfs_close(out_v);
    UpdateList = StreamPrintEnd();
    DynaEvalString("(compile-file \"init.scm\" \"init.dat\")",&type);

    // Update DynaMulti
    UpdateMultiFile(MakeDquteString(DllReletivePath),
                    MakeDquteString(appname),
                    UpdateList),
    WriteMulti(); // Write DynaMulti data

    // Wirte Ship File
    CreateShipFile(argv[1],appname,dllpath,dllname,UpdateList);

  }
  else { /* Only create VFS */
    ret_v = CFSMountWithPath(appath);
    switch(ret_v) {
    case 0· /* Mount Success */
      break;
    case 1· /* Wrong Path Name Use */
      printf("Wrong path Name use in %s  Must have .exe \n",appath);
      exit(-1);
    case 2  /* Different VFS use */
        printf("Wrong VFS Use \n");
        exit(-1);
    case 3: /* Does not exits VFS */
      if(CreateCFS(appath)) {
        printf("Access denied in %s \n",appath);
        exit(-1);
      }
```

```
            break;
        }
    }
    exit(0);
}
```

```
/*
 * cfs h
 *
 */

#ifndef _CFS_H_
#define _CFS_H_

#include    <sys/types.h>
#include    <time.h>

/*
 * absorbation difference between K&R C and ANSI C
 *
 */

#ifndef const
#ifndef __STDC__
 #define const
 #endif /* __STDC__ */
#endif /* const */

#ifndef __P
#ifdef __STDC__
#define __P(protos) protos      /* for ANSI C prototype */
#else
#define __P(protos) ()      /* for K&R C prototype  */
#endif  /* __STDC__ */
#endif  /* __P */

/*
 * system dependent type definition
 *
 */
#if (defined solaris)
typedef unsigned char   uchar;
#elif   (defined sun)
typedef unsigned char   uchar,
typedef unsigned long   ulong,
#elif   (defined __FreeBSD__)
typedef unsigned long   ulong;
typedef unsigned char   uchar;
#elif   (defined linux)
typedef unsigned char   uchar;
#else
typedef unsigned int    uint,
typedef unsigned short  ushort;
typedef unsigned long   ulong,
typedef unsigned char   uchar;
#endif

/*
 * defines
 *
 */

#define CFS_FNMAX   256     /* filename maximum length  */
#define CFS_PNMAX   1024        /* pathname maximum length  */
#define CFS_PDMAX   256     /* path depth max        */
#define CFS_ANMAX   CFS_FNMAX   /* aspectname maximum length    */

#define CEOF        (-1)

/* for file seek */
#define CFS_SEEK_SET    0
#define CFS_SEEK_CUR    1
#define CFS_SEEK_END    2

/* for file open mode */
#define CO_RDONLY   0x0001
#define CO_WRONLY   0x0002
#define CO_APPEND   0x0004
#define CO_RDWR     0x0008
/*
 * structure definition
 *
```

```
 */
struct cfs_rsb,
struct cfs_rbg;
struct cfs_sb;
struct cfs_bg,
struct cfs_cfile;   /* real definition is in cfs_file.h */
struct cfs_dir;     /* real definition is in cfs_dir.h */

/* cnode definition */
#define     NDBLK   16
#define     NSIBLK  4
#define     NDIBLK  2
#define     NTIBLK  2
struct rcnode {
    short       nref;       /*   0: rcnode reference count  */
    unsigned short  mode;       /*   2: file attribute      */
    time_t      atime;      /*   4: access time     */
    time_t      mtime;      /*   8. modified time       */
    time_t      ctime;      /*  12: cnode change time   */
    long        length,     /*  16  file length     */
    long        blocks,     /*  20: using blocks        */
    unsigned long   d_db[NDBLK];     /*  24· direct disk block   */
    unsigned long   si_db[NSIBLK];  /*  88: single indirect db  */
    unsigned long   di_db[NDIBLK];  /* 104: double indirect db  */
    unsigned long   ti_db[NTIBLK];  /* 112· triple indirect db  */
    long        spare[2];   /* 120: reserved        */
};

/* constant for rcnode mode */
#define RCN_DIR     0x8000      /* directory            */
#define RCN_RF      0x4000      /* regular file         */
#define RCN_PM      0x2000      /* polymorph state      */
#define RCN_PF      (RCN_PM | RCN_DIR)  /* polymorph file   */
#define RCN_AF      (RCN_PM | RCN_RF)   /* aspect file      */

#define RCN_READ    0x0100      /* file readable        */
#define RCN_WRITE   0x0080      /* file writable        */
#define RCN_EXECUTE 0x0040      /* file executable      */

struct cnode {
    struct cnode    *prev;      /* previous cnode in list   */
    struct cnode    *next;      /* next cnode in list       */
    int     fsn;        /* file system number       */
    unsigned long   cnn;        /* cnode number         */
    short       nref;       /* cnode reference count    */
    short       nlock,      /* lock count           */
    long        il;     /* indirect level(not used yet) */
    long        lblocks,    /* number of logical blocks */
    unsigned long   flags;      /* flags            */
    struct rcnode   rcn;        /* on-disk cnode data       */
},

/* flags for struct cnode */
#define CN_USED     0x0001
#define CN_SHLOCK   0x0002
#define CN_EXLOCK   0x0004
#define CN_ACCESS   0x0008
#define CN_CHANGE   0x0010
#define CN_MODIFY   0x0020

/* directory entory definition */
struct cdent {
    unsigned long   cnn,        /* cnode number         */
    unsigned short  entlen;     /* length of this dir entry */
    unsigned char   type;       /* file type (not used yet) */
    unsigned char   cfnlen,     /* length of filename       */
    char        cfname[CFS_FNMAX];  /* file name        */
},

/* constant for cfs_lock */
#define CL_UN   0x0000
#define CL_SH   0x0001
#define CL_EX   0x0002

/*
 * type definition
 *
 */
typedef struct cnode        CN;
typedef struct cdent        CDENT;
typedef struct cfs_cfile    CF;
typedef struct cfs_dir      CDIR;


/*
 * prototype declaration
 *
 */

/* cfs_cfsio.c */
extern CF   *cfs_open           __P((const char *cpath, char mode)),
extern void cfs_close           __P((CF *cfp));
extern int  cfs_read            __P((void *buf, size_t size,
                  size_t nmemb, CF *cfp));
  tern int  cfs_write           __P((const void *buf, size_t size,
                  size_t nmemb, CF *cfp));
extern void cfs_truncate        __P((CF *cfp)),
extern int  cfs_flush           __P((CF *cfp));

/* cfs_cnode */
extern int  cfs_cnode           __P((const char *cpath, CN *cn));

/* cfs_crypt c */
extern int  cfs_decode_read     __P((void *buf, size_t csize,
                  size_t cnmemb, CF *cfp)),
extern int  cfs_encode_write    __P((const void *buf, size_t bsize,
                  size_t bnmemb, CF *cfp));
/* cfs_dir c */
extern int  cfs_mkdir           __P((const char *cpath, ushort mode));
extern int  cfs_rmdir           __P((const char *cpath));
extern int  cfs_link            __P((const char *opath, const char *npath));
extern int  cfs_remove          __P((const char *cpath)),
extern int  cfs_rename          __P((const char *opath, const char *npath));
extern int  cfs_chdir           __P((const char *cpath));
extern CDIR *cfs_opendir        __P((const char *cpath));
extern void cfs_closedir        __P((CDIR *cdp));
extern CDENT    *cfs_readdir        __P((CDIR *cdp));
extern void cfs_seekdir         __P((CDIR *cdp, long location));
extern void cfs_rewinddir       __P((CDIR *cdp));
extern long cfs_telldir         __P((CDIR *cdp));

/* cfs_file.c */
extern int  cfs_fseek           __P((CF *cfp, long offset, int whence));
extern long cfs_ftell           __P((CF *cfp));
extern void cfs_rewind          __P((CF *cfp));
extern int  cfs_eof             __P((CF *cfp));
extern int  cfs_error           __P((CF *cfp));
extern void cfs_clear_eof       __P((CF *cfp));

   cfs_init.c */
extern int  cfs_make_new_fs     __P((const char *pathname, long bs,
                    long ts, long cco));
/* cfs_lock.c */
extern int  cfs_lock            __P((CF *cfp, int op));
extern int  cfs_locktype        __P((CF *cfp));
extern int  cfs_has_lock        __P((CF *cfp));

/* cfs_mount.c */
extern int  cfs_mount           __P((const char *pathname)),
extern int  cfs_umount          __P((const char *pathname));

/* cfs_polymorph c */
extern int  cfs_declare_aspect  __P((const char *aspect));
extern void cfs_clear_aspect    __P((void));
extern char *cfs_current_aspect __P((void));

#endif  /* _CFS_H_ */
```

```
/*****************************************************************
  directory c  (Dynagen function)
       By Takashi Kosaka (C) SegaSoft INC.    1997 -

  SEGASOFT CONFIDENTIAL - Unpublished Copyright (c) [1997],
  SegaSoft, Inc.  All Rights Reserved.
******************************************************************/

#include <stdio.h>
#define WINDOWS
#ifdef WINDOWS
#include <windows.h>
#endif

typedef struct _MyDir {
  char *Dir;
  struct _MyDir *Next;
` MyDir, *PMyDir;

   /Dir CreateDirectroyData(char *path);

PMyDir CreateDirectroyData(char *path)
{
  PMyDir top,now,old;
  int i,len,prv;
  char *pp,

  old = top = (PMyDir)NULL;
  len = strlen(path);

  for( i = 0 ; i < len ; i++ ) {
    if(*(path + i) == '·') {
      pp = path + i + 2;
      break;
    }
  }

  len = strlen(pp);
  prv = 0;
  for( i = 0 , i < len ; i++ ) {
    if(*(pp + i) == '/' || *(pp + i) == '\\') {
      if((now = (PMyDir)malloc(sizeof(MyDir))) == NULL) {
        fprintf(stderr,"Error· Can not allocate memroy %d bytes\n",
        sizeof(MyDir));
        exit(-1);
      }
      if(!top) top = now;
      now->Dir = (pp + prv);
      now->Next = (PMyDir)NULL;
      *(pp + i) = '\0';
      // printf("Dir: %s \n",now->Dir);
      if(old){
        old->Next = now,
      }
      old = now;
      prv = i + 1;
    }
  }
  return((PMyDir)top);
}

// String copy by lower case
void StrCpyLowcase(char *dst, char *src)
{
  for( ; *src ; src++) {
    if(*src >= 'A' && *src <= 'Z')
      *dst++ = *src - 'A' + 'a',
    else
      *dst++ = *src,
  }
}

#define OneDown(now)  ((now)->Next)
#define DirNotSame(src,dst)  (strcmp((src)->Dir,(dst)->Dir))

// Create Relative Path
```

```
char * CreateReletivePath(char *src_path. char *dst_path)
{
  static char SrcDir[2048],DstDir[2048];
  static char Target[1024];
  PMyDir src,dst,snow,dnow;
  int len,i;

  StrCpyLowcase(SrcDir,src_path);
  StrCpyLowcase(DstDir,dst_path);

  // Drive Check
  if(SrcDir[0] != DstDir[0])  {
    return((char *)DstDir);
  }

  src = CreateDirectroyData(SrcDir);
  dst = CreateDirectroyData(DstDir);

  dnow = dst;
  Target[0] = '\0';
  for( snow = src ; snow ; snow = OneDown(snow)) {
    if(DirNotSame(snow,dnow)) {
      while(snow) {
    strcat(Target,"  /"),
    snow = OneDown(snow);
      }
      break;
    }
    dnow = OneDown(dnow);
    if(!dnow) {
      snow = OneDown(snow);
      while(snow) {
    strcat(Target,"../");
    snow = OneDown(snow);
      }
      break;
    }
  }

  while(dnow) {
    strcat(Target,dnow->Dir);
    strcat(Target,"/"),
    dnow = OneDown(dnow);
  }
  len = strlen(dst_path) - 1,
  for( i = len , i > 0 ; i-- ) {
    if(*(dst_path + i) == '/' ||
       *(dst_path + i) == '\\') {
      strcat(Target,dst_path + i + 1);
      break;
    }
  }
  return((char *)Target);
}

/* Make Application Path name */
void MakeAppPath(char *DefFilePath,char *AppName,char *AppPath)
{
  int i,len;

  strcpy(AppPath,DefFilePath),
  len = strlen(AppPath) - 1;
  for( i = len , i > 0 ; i-- ) {
      if(*(AppPath + i) == '/' ||
         *(AppPath + i) == '\\' ) {
          strcpy(AppPath + i + 1,AppName);
          break;
      }
  }
}

// AppName has "appname"
// This function take out "
void GetRealAPPName(char *appname)
{
  int i,len,flg;
```

```
char buf[1024];
flg = 0,
len = strlen(appname);
for(ı = 0 ; ı < len ; ı++ ) (
  ıf(*(appname + ı) == '*') (
    ıf('flg)
      flg = ı + 1;
    else
      *(appname + ı) = '\0';
  }
}
strcpy(buf,appname + flg);

len = strlen(buf);
/* Make Down Case strıng */
for(ı = 0 , ı < len ; ı++ ) (
    ıf(buf[ı] >= 'A' && buf[ı] <= 'Z')
         buf[ı] = buf[ı] - 'A' + 'a',


 strcpy(appname,buf);
)
```

```
/*****************************************************************
  ship c  (Dynagen function)
     By Takashi Kosaka (C) SegaSoft INC.    1997 -

  SEGASOFT CONFIDENTIAL - Unpublished Copyright (c) [1997],
  SegaSoft, Inc.  All Rights Reserved
*****************************************************************/

#include <stdio h>
#include <time h>
#define WINDOWS
#ifdef WINDOWS
#include <windows.h>
#endif

#define APP "/app "
#define DYNAMODULE "/dynamodule:"
#define INITSC "/initsc:"
 define VFS     "/vfs:"
 efine VFSDATA "/vfsdata:"
 efine DATA "/data:"

void CreateShipFile(char *AppDefPath,char *AppName,char *DllPath,char *DllName,char *UpdateList)
{
    char TergetDLLPath[1024];
    char ShipPath[1024];
    char TimeBuf[1024];
    FILE *fp;
    int i,len;
    time_t ltime;

    time( &ltime );

    // Create Ship File Name
    strcpy(ShipPath,AppDefPath);
    len = strlen(ShipPath) - 1;
    for( i = len ; i > 0 ; i-- ) {
        if(*(ShipPath + i) == '/' ||
          *(ShipPath + i) == '\\' ) {
            strcpy(ShipPath + i + 1,DllName);
            break,
        }
    }
    len = strlen(ShipPath),
    strcpy(ShipPath + len - 3,"Shp"),

    len = strlen(DllName);
    strcpy(TergetDLLPath,DllName);
    strcpy(TergetDLLPath + len - 4,"/");

    if((fp = fopen(ShipPath,"w")) == NULL) {
        fprintf(stderr,"Error: Can Not Create %s file \n",ShipPath);
        exit(-1);
    }
    strcpy(TimeBuf,ctime( &ltime )),
    for( i = 0 , TimeBuf[i] , i++ ) {
        if(TimeBuf[i] == '\n')
            TimeBuf[i] = '\0',
    }
    fprintf(fp,"; ################################################################ \n"),
    fprintf(fp,", #             This is %s Ship File                            \n",DllName),
    fprintf(fp,"; # Created by Dynagen.exe  Date: %s \n",TimeBuf),
    fprintf(fp,", ################################################################ \n"),
    fprintf(fp,"%s%s\n",APP,AppName),
    fprintf(fp,"%s%s|%s%s\n",DYNAMODULE,DllPath,TergetDLLPath,DllName),
    fprintf(fp,"/initscname:%s\n",DllName);
    fprintf(fp,"%s(enable-dynamod \"%s%s\" \"%s\" \\\n%s)\n\n",INITSC,TergetDLLPath,DllName,AppNam
e,UpdateList);
    fprintf(fp,"; ################################################################ \n");
    fprintf(fp,"; #             Following commands You can Modify               # \n"),
    fprintf(fp,"; # * If you want to modify commands, you have to take          # \n");
    fprintf(fp,"; #   out ';' at first line.                                    # \n");
    fprintf(fp,"; # * The root of a terget path is the directory where          # \n");
    fprintf(fp,"; #   is an application in user environmant.                    # \n");
    fprintf(fp,"; # * /initscb is a control script that represents before       # \n");
    fprintf(fp,"; #   before /initsc. /initsca means a control script that      # \n");
```

```
    fprintf(fp,"; #   represents after /initsc.                                 # \n");
    fprintf(fp,"; # * /initscb + /initsc + /initsca is a control script for# \n");
    fprintf(fp,"; #   DynaModule. The control script saves XXX.dat              # \n");
    fprintf(fp,"; # * /dependent:          is a dependent modules which is      # \n"),
    fprintf(fp,", #   used by this DynaModule.                                  # \n"),
    fprintf(fp,"; # * /eval  is the evaluating script in a DynaInstall exe # \n");
    fprintf(fp,"; # * Use '\\' and CR to continure a line                          # \n"),
    fprintf(fp,"; ################################################################ \n");
    fprintf(fp,";/scinitb· ControlScript_brefore_'initsc' \n");
    fprintf(fp,",/scinita. ControlScript_after_'initsc' \n");
    fprintf(fp,";/dependent·module_name1,module_name2,... \n");
    fprintf(fp,";/eval ControlScript \n"),
    fprintf(fp,";%sSource_path|Target_path_in_VFS\n",VFS);
    fprintf(fp,";%sValue|Target_path_in_VFS\n",VFSDATA);
    fprintf(fp,";%sSource_path|Releative_Target_path.\n",DATA);
    fclose(fp);
}
```

```
/***************************************************************
 StoreMulti.c  (Dynagen function)
      By Takashi Kosaka (C) SegaSoft INC     1997 -

 SEGASOFT CONFIDENTIAL - Unpublished Copyright (c) [1997],
 SegaSoft, Inc   All Rights Reserved
****************************************************************/

#include <stdio.h>
#define WINDOWS
#ifdef WINDOWS
#include <windows.h>
#endif

// Function Definition
char *FindFunctionNameFromNumber(char *number);

char *FindNumberFromList(char *list,char *number)

    int i,len,size;
    char *ret,*next;
    size = 0;
    len = strlen(list),
    ret = (char *)NULL;
    for( i = 0 ; i < len ; i++ ) {
        if( *(list + i) == '(' ) {
            i++;
            if(*(list + i) == '(')
                ret = list + ++i;
            else
                ret = list + i;;
            size ++;
        }
        else if(*(list + i) == ' ' && ret) {
            next = list + i + 1;
            break;
        } else if(ret)
            size++;
    }
    if(i >= len)
        return((char *)NULL),

    for(i = 0 , i < size ; i++ )
        *(number + i) = *(ret + i),

    *(number + i) = '\0';

    return((char *)next),
}

char *GetNameFromPath(char *path)
{
    int len,i;
    len = strlen(path) - 1;
    for( i = len ; i > 0 ; i-- ) {
        if(*(path + i) == '/' ||
           *(path + i) == '\\' ) {
            return((char *)path + i + 1);
        }
    }
    return((char *)path + 1);
}

// option: 0, Constructor
// option· 1, Desctructor
// opetin  3, Member Function
// Terminate is '@@'
void GetFunNameClassName(char *src,char *dst,int option)
{
    char Class[512],Fname[512];
    int i,len,flg;
    len = strlen(src);
    flg = -1,
    Class[0] = '\0';
    for( i = 0 ; i < len , i++ ) {
        if(*(src + i) == '@' ) {
```

```
            if(*(src + i + 1) == '@') {
                Class[flg] = '\0';
                if(i < 512)
                    Fname[i] = '\0',
                break,
            } else {
                flg++,
                Fname[i] = '\0';
            }
        }
        else if(flg >= 0)
            Class[flg++] = *(src + i);
        else
            Fname[i] = *(src + i);
    }
    switch(option) {
    case 0: // Constructor
        sprintf(dst,"%s::%s",Fname,Fname);
        break;
    case 1:
        sprintf(dst,"~%s:·%s",Fname,Fname);
        break;
    default:
        if(Class[0])
            sprintf(dst,"%s::%s",Class,Fname);
        else
            sprintf(dst,"%s",Fname);
        break;
    }
}

char *MakeReadableFunctionName(char *fname)
{
    static char buf[1024];

    if(!fname)
        return((char *)NULL);

    if(strncmp("??0",fname,3) == 0) { // Constructor
        GetFunNameClassName(fname + 3,buf,0);
    }
    else if(strncmp("??1",fname,3) == 0) { // Destructor
        GetFunNameClassName(fname + 3,buf,1);
    }
    else if (strncmp("??_GC",fname,5) == 0) {
        return((char *)NULL);
    }
    else if(*fname == '?') { // Member Function or ANSI C Function !!!!!
        GetFunNameClassName(fname + 1,buf,2),
    }
    else { // C Function
        strcpy(buf,fname + 1);
    }
    return((char *)buf);
}


void CheckMultipleUse(char *OldDllPath,char *oldlist,char *newlist)
{
    char oldnum[256],newnum[256],*oldnext,*newnext,
    char OldDllName[256];
    char *Fname;
    int i;

    strcpy(OldDllName,GetNameFromPath(OldDllPath));
    for( i = 0 , OldDllName[i] , i++) {
        if(OldDllName[i] == '.') {
            OldDllName[i] = '\0';
            break;
        }
    }

    for(oldnext = oldlist , oldnext ;  ) {
        oldnext = FindNumberFromList(oldnext,oldnum);
        if(!oldnext)
            break,
        for(newnext = newlist ; newnext ; ) {
            newnext = FindNumberFromList(newnext,newnum);
            if(!newnext)
                break;
            if(strcmp(oldnum,newnum) == 0) {
                Fname = MakeReadableFunctionName(FindFunctionNameFromNumber(oldnum));
                if(Fname) {
                    fprintf(stderr,"Warning. %s has already used in %s. \n",
                            Fname,OldDllName);
                }
                break;
            }
        }
    }
}

// StreamMemory
typedef struct _StreamMem {
    int size,
    unsigned char *mem;
    int offset;
    int init,
} Stream, *PStream;

static Stream TopStream = {0,(unsigned char *)NULL,0,1};

#define STREAMSIZE 1024

// data into a Stream
void PushByte(unsigned char data)
{
    if(TopStream init) {
        if((TopStream.mem =
            (unsigned char *)malloc(STREAMSIZE)) == NULL) {
            fprintf(stderr,"Error. Can not allocate Memory %d \n",STREAMSIZE);
        }
        TopStream.size = STREAMSIZE;
        TopStream.init = 0;
    }
    else if(TopStream offset >= TopStream.size ) {
        unsigned char *tmp,
        if((tmp = (unsigned char *)malloc(TopStream.size + STREAMSIZE)) == NULL) {
            fprintf(stderr,"Error: Can not allocate Memory %d \n", TopStream.size + STREAMSIZE);
            exit(-1);
        }
        memcpy(tmp,TopStream mem,TopStream.size),
        free(TopStream mem),
        TopStream mem = tmp,
        TopStream.size += STREAMSIZE;
    }
    *(TopStream.mem + TopStream offset++) = data;
}

   Multi data into a stream
// Data must has NULL terminate
void PushBytes(unsigned char *data)
{
    while(*data) {
        PushByte(*data++);
    }
}

// Get buffer from a Stream
unsigned char *GetStreamBuffer(int *size)
{
    if(TopStream offset) {
        unsigned char *tmp;
        *size = TopStream offset,
        if((tmp = (unsigned char *)malloc(*size)) == NULL) {
            fprintf(stderr,"Error: Can not allocate Memory %d bytes \n",*size),
            exit(-1);
        }

        memcpy(tmp,TopStream.mem,*size),
        TopStream offset = 0,
```

```
        return((unsigned char *)tmp);
    }
    else { // None
        *size = 0,
        return((unsigned char *)NULL);
    }
}

// Multi Dynamodule Handle structure
typedef struct _Multi {
    char *ModulePath;
    char *AppName;
    char *UpdateList;
    struct _Multi *next;
} Multi, *PMulti;

static PMulti TOP = (PMulti)NULL;
static PMulti OLD = (PMulti)NULL;
/*********************************************************
* Line Representation
* "DynaModule-Path" "Application Name" '((12 . #x20) ... .
*********************************************************/
/* Set Mulit Data */
void SetMultiData(char *line,int len)
{
    int i,first;
    PMulti now;

    if(!line)
        return;

    if((now = (PMulti)malloc(sizeof(Multi))) == NULL) {
        fprintf(stderr,"Can not allocate memory %d bytes\n",
            sizeof(Multi));
        exit(-1);
    }
    if(!TOP) TOP = now;
    if(OLD) OLD->next = now;
    OLD = now;
    now->next = (PMulti)NULL;
    now->ModulePath = (char *)NULL;
    now->AppName = (char *)NULL;
    now->UpdateList = (char *)NULL;
    first = 1;
    for( i = 0 ; i < len ; i++ ) {
        if(*(line + i) == '"' && first ) {
            if(!now->ModulePath)
                now->ModulePath = line + i;
            else if(!now->AppName)
                now->AppName = line + i;
            first = 0;
        }
        else if (*(line + i) == '"' && !first) {
            first = 1;
        }
        else if (*(line + i) == '\'') {
            now->UpdateList = line + i;
            break;
        }
        else if (*(line + i) == ' ' && first) {
            *(line + i) = '\0';
        }
    }
}

void SkipUntilCr(FILE *fp)
{
    int data;
    data = fgetc(fp),
    while(data != '\n')
        ;
}

void UpdateMultiFile(char *DllPath,char *AppName, char *UpdateList)
{
    PMulti now,last,
    int set;
    set = 0,
    last = (PMulti)NULL;
    for( now = TOP ; now ; now = now->next) {
        if(strcmp(now->ModulePath,DllPath) == 0 &&
           strcmp(now->AppName,AppName) == 0) {
            now->UpdateList = UpdateList;
            set = 1;
            break;
        }
        else {
            CheckMultipleUse(now->ModulePath,now->UpdateList,UpdateList),
        }
        last = now,
    }
    if(!set) { // Add New
        if((now = (PMulti)malloc(sizeof(Multi))) == NULL) {
            fprintf(stderr,"Can not allocate memory %d bytes\n",
                sizeof(Multi));
            exit(-1);
        }
        now->next = (PMulti)NULL;
        now->ModulePath = DllPath;
        now->AppName = AppName;
        now->UpdateList = UpdateList;
        if(last) last->next = now;
        if(!TOP) TOP = now;
    }
}

static char dynamulti[1024]; // path name

// Read Multi Dynamodule control File
void ReadMultiFile(char *app_def_path)
{
    char *line;
    FILE *fp;
    int data,len,code;

    strcpy(dynamulti,app_def_path),
    len = strlen(dynamulti),
    dynamulti[len - 3] = '\0';
    strcat(dynamulti,"mul"),

    if((fp = fopen(dynamulti,"r")) == NULL) {
        return,
    }
    code = 0,
    // Reading Data
    for( ; ; ) {
        data = fgetc(fp);
        if(data == EOF)
            break;
        else if(data == '#') {
            code = fgetc(fp),
            if(code == '(')
                break;
            else {
                PushByte((unsigned char)data);
                PushByte((unsigned char)code),
                code = 0,
            }
        }
        else if (data == '\\') { // continue check
            code = fgetc(fp);
            if(code != '\n') { // continue
                PushByte((unsigned char)data);
                PushByte((unsigned char)code),
                code = 0;
            }
        }
        else if (data == ',')
            SkipUntilCr(fp);
        else if(data == '\n') {
            PushByte('\0'),
            line = GetStreamBuffer(&len),
```

```
                SetMultiData(line,len);
            }
            else
                PushByte((unsigned char)data),
        }
        if(!code) {
            line = GetStreamBuffer(&len);
            SetMultiData(line,len),
        }
        else
            GetStreamBuffer(&len);

        fclose(fp);
}

// Write Multi DynaModule data file
void WriteMulti()
{
    PMulti now;
    FILE *fp;

    if((fp = fopen(dynamulti,"w")) == NULL) {
        fprintf(stderr,"Error: Can not Create File %s \n",dynamulti);
        exit(-1);
    }
    for( now = TOP ; now ; now = now->next) {
        fprintf(fp,"%s %s \\\n%s\n",now->ModulePath,
                now->AppName,now->UpdateList);
    }
    fclose(fp),
}


// Make String with "xxxxx"
char *MakeDquteString(char *string)
{
    int size;
    PushByte('"');
    PushBytes(string);
    PushByte('"');
    PushByte('\0');
    return((char *)GetStreamBuffer(&size));
}

// Stream Printf
void StreamPrintf(char *format,char *arg1, char *arg2)
{
    char buf[2048];
    int i;

    for(i = 0 , i < 2048 , i++ )
        buf[i] = '\0';

    if(arg1)
        sprintf(buf,format,arg1,arg2);
    else
        sprintf(buf,format);

    PushBytes(buf);
}

char * StreamPrintEnd()
{
    int size;
    PushByte('\0');
    return((char *)GetStreamBuffer(&size)),

}
```

DynaObj EXE

```
/* MakePtr is a macro that allows you to easily add to values (including
// pointers) together without dealing with C's pointer arithmetic.  It
// essentially treats the last two parameters as DWORDs.  The first
// parameter is used to typecast the result to the appropriate pointer type  */

#define MakePtr( cast, ptr, addValue ) (cast)( (unsigned char *)(ptr) + (addValue) )

typedef struct {
   unsigned char * stringTable,
   PIMAGE_FILE_HEADER pImageFileHeader;
   int COFFSymbolCount;
   PIMAGE_SYMBOL PCOFFSymbolTable;
   int size;
   /* For Orignal COFF */
   PSYMBOL_CHAIN topsymbol,
   PSYMBOL_CHAIN oldsymbol;
   PSTRING_T_CHAIN topstring;
   PSTRING_T_CHAIN oldstring;
   int NewStringTableSize;
   int NextSymbolNumber;
   int NextStringIndex;
   long EndOfSymbolTable;
 } CONTROL;

 extern CONTROL ocontrol;
```

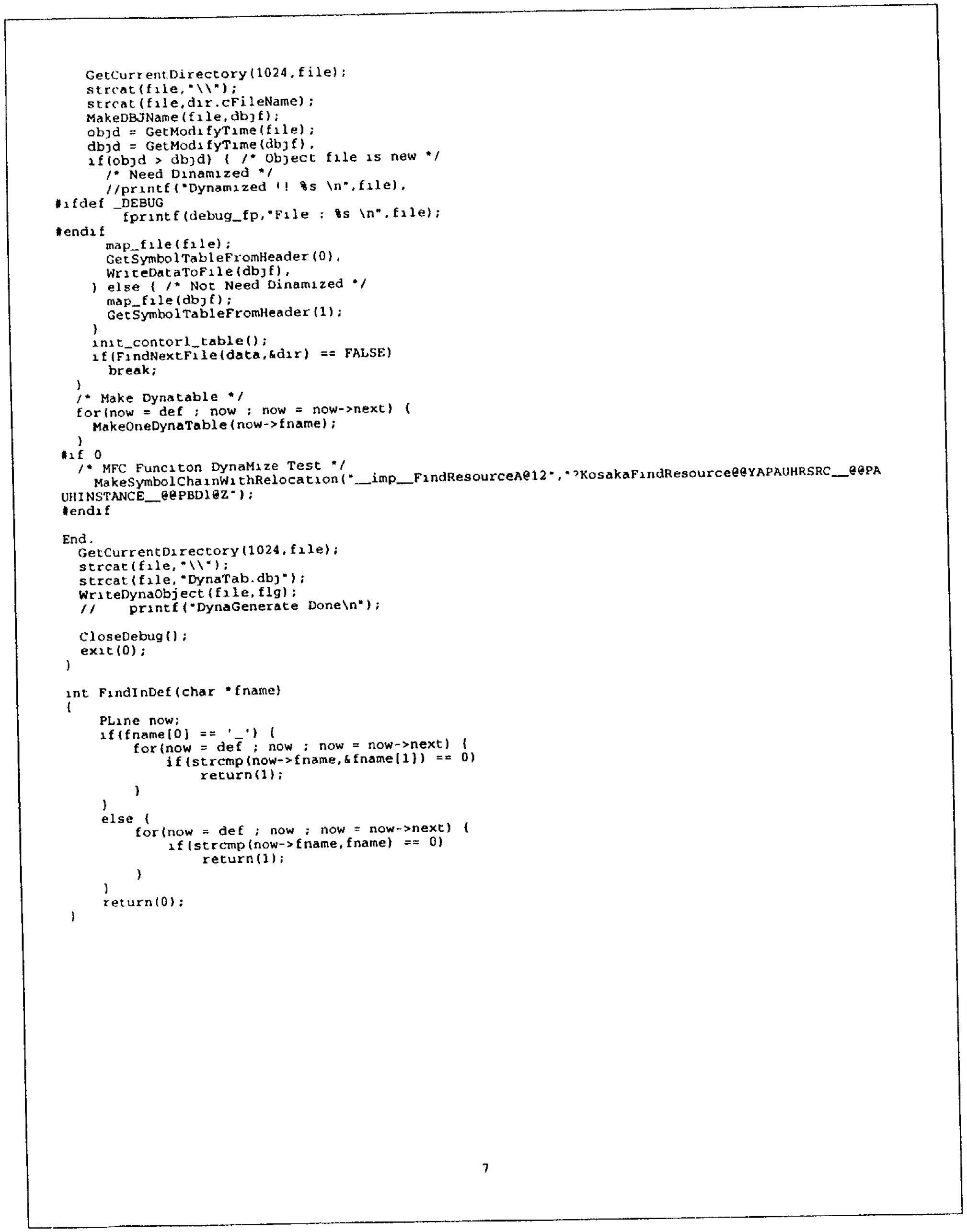
```
/*****************************************************
DynaObj:  SegaSoft NetWork Inc. (c) by Takashi Koska
    This Program changes obj file to dbj file and
    creates dynatab.dbj.
*****************************************************/

#include <stdio h>
#define WINDOWS
#ifdef WINDOWS
#include <windows.h>
#include <string h>
#include <time h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#else
#include "winnt.h"
*endif

   def _DEBUG
rILE *debug_fp,
#endif

// Debug Statement Open
void OpenDebug()
{
#ifdef _DEBUG
    if((debug_fp = fopen("OBJdebug.txt","w")) == NULL)
        exit(-1);
#endif
}

// Debug Statement Close
void CloseDebug()
{
#ifdef _DEBUG
    fclose(debug_fp);
#endif
}


typedef struct _symbol_chain {
  PIMAGE_SYMBOL symbol;
  struct _symbol_chain *next;
} SYMBOL_CHAIN, *PSYMBOL_CHAIN,

typedef struct _string_t_chain {
  char *name;
  struct _string_t_chain *next;
} STRING_T_CHAIN, *PSTRING_T_CHAIN;

typedef struct {
   unsigned char * stringTable;
   PIMAGE_FILE_HEADER pImageFileHeader;
   int COFFSymbolCount;
   PIMAGE_SYMBOL PCOFFSymbolTable;
   int size;
   /* For Orignal COFF */
   PSYMBOL_CHAIN topsymbol;
   PSYMBOL_CHAIN oldsymbol;
   PSTRING_T_CHAIN topstring;
   PSTRING_T_CHAIN oldstring,
   int NewStringTableSize;
   int NextSymbolNumber;
   int NextStringIndex;
   long EndOfSymbolTable;
 } CONTROL;

CONTROL ocontrol = ((unsigned char *)NULL,
            (PIMAGE_FILE_HEADER)NULL,
             0,(PIMAGE_SYMBOL)NULL,0,
            (PSYMBOL_CHAIN)NULL,(PSYMBOL_CHAIN)NULL,
            (PSTRING_T_CHAIN)NULL,(PSTRING_T_CHAIN)NULL,
            0,0,0,0);

int map_file(char *file_name);
```

```
int GetDynaPlaySymbolIndex(PIMAGE_SYMBOL pSymbolTable ,int cSymbols);
int search_string(char * source,char *target);
void DynaMaizedAPP();
void DynaModAPP();
int GetSymbolTableFromHeader(int flg),
void WriteDynaObject(char *name,int flag);
void WriteDataToFile(char *file_name);
char *stralloc(char *buf);
void MakeSymbolChainWithRelocation(char *org_name,char *new_name),

void FreeNewSymbolTable()
{
  PSYMBOL_CHAIN now;
  PSYMBOL_CHAIN old;

  for(now = ocontrol.topsymbol ; now ;) {
    free(now->symbol);
    old = now,
    now = now->next;
    free(old);
  }
}

void FreeNewStringTable()
{
  PSTRING_T_CHAIN now;
  PSTRING_T_CHAIN old;

  for(now = ocontrol.topstring ; now ;) {
    free(now->name);
    old = now;
    now = now->next;
    free(old);
  }
}

void init_contorl_table()
{
  ocontrol.stringTable = (unsigned char *)NULL;
  free(ocontrol pImageFileHeader);
  ocontrol.pImageFileHeader = (PIMAGE_FILE_HEADER)NULL;
  ocontrol.COFFSymbolCount = 0,
  ocontrol.PCOFFSymbolTable = (PIMAGE_SYMBOL)NULL;
  ocontrol.size = 0;
  FreeNewSymbolTable();
  FreeNewStringTable();
  ocontrol.topsymbol = (PSYMBOL_CHAIN)NULL;
  ocontrol.oldsymbol = (PSYMBOL_CHAIN)NULL;
  ocontrol.topstring = (PSTRING_T_CHAIN)NULL;
  ocontrol oldstring = (PSTRING_T_CHAIN)NULL;
}


void MakeDBJName(char *path,char *dst)
{
  int i,len,
  len = strlen(path),
  strcpy(dst,path),
  for(i = len , i > 0 , i-- ) {
    if(*(dst + i) == ' ') {
      strcpy(dst + i ,".dbj");
      break,
    }
  }
}

unsigned long GetModifyTime(char *path)
{
  struct _stat buf,
  if(_stat(path,&buf) < 0)
    return((unsigned long)0);
  else
    return((unsigned long)buf st_mtime);
}

void ReplaceDirectoryName(char *buf)
```

```
{
  int len,i,
  len = strlen(buf),
  for(i = 0 ; i < len , i++ ) {
    if(*(buf + i) == '\\')
      *(buf + i) = '/';
  }
}

char app_name[1024];

/*****************************************************************
  DEF file interface
*****************************************************************/
/* Line management Table in .def file */
typedef struct _Line {
  char *fname,
  int no,
  int option;
  char *add;
  struct _Line *next;
} Line, *PLine;


PLine SearchLine(char *func,PLine top)
{
  PLine now;
  for(now = top ; now ; now = now->next) {
    if(strcmp(func,now->fname) == 0)
      return((PLine)now);
  }
  return((PLine)NULL);
}

/* Read One Line */
int ReadLine(FILE *fp,char buf[],int max_size)
{
  int i,data;
  for(i = 0 , i < max_size ; i++ ) {
    data = getc(fp);
    if(data == EOF) {
      buf[i] = '\0';
      return(EOF);
    }
    else if(data == '\n') {
      buf[i] = '\0';
      return(0);
    }

    buf[i] = data;
  }
  buf[i - 1] = '\0';
  return(i);



/* Set correct data into Line Data. This is sequentail setting */
void SetIntoLine(char *src,PLine line,int no)
{
  if(!line->fname)
    line->fname = stralloc(src);
  else if (!line->no) {
    line->no = no;
  }else if(!line->option)
    line->option = 1;
  else if(!line->add) {
    if(*src == '0')
      *src = '#';
    line->add = stralloc(src).
  }
}

// Find = in Def file statement
int FindEQ(char *data)
{
    int i;
```

```
    for( i = 0 ; *(data + i) ; i++ ) {
        if(*(data + i) == '=') {
            if(*(data + i + 1) == '_')
                return(i + 2);
            else
                return(i + 1);
        }
    }
    return(0);
}

/* Create Line Data from String */
PLine CreateLineData(char *data,int no)
{
  int i,len,code;
  char *start;

  PLine now;
  if((now = (PLine)malloc(sizeof(Line))) == NULL) {
    printf("Can not allocate Line memory !!!\n");
#ifdef _DEBUG
    fprintf(debug_fp,"Can not allocate Line Memory \n");
#endif
    CloseDebug();
    exit(1);
  }
  now->fname = (char *)NULL,
  now->no     = 0;
  now->option = 0;
  now->add    = (char *)NULL;
  now->next   = (PLine)NULL;
  code = 0;
  len = strlen(data);
  data += FindEQ(data);
  start = data;
  for(i = 0 ; i < len + 1; i++ ) {
    if((*(data + i) != ' ' || *(data + i) == '\0' || *(data + i) != '    ')
        && code == 0) {
      code = 1;
      start = data + i;
    }
    if(code == 1 && (*(data + i) == ' ' || *(data + i) == '\0' || *(data + i) == '    ')) {
      code = 0;
      *(data + i) = '\0';
      SetIntoLine(start,now,no);
    }
  }
#ifdef _DEBUG
  fprintf(debug_fp,"%s \n",now->fname);
#endif
  return((PLine)now);
}

static int NumberOfFunction = 0,

/* Load def into memory */
PLine  LoadDefFileIntoMemory(FILE *fp)
{
  PLine top,now,old;
  char buf[2048],
  int err,no;

  no = 0;
  old = top = (PLine)NULL;
  err = ReadLine(fp,buf,2048),
  while (err != EOF) {
    if(buf[0] == ',')
      break,
    if((now = CreateLineData(buf,no++))) {
      if(old) old->next = now,
      if(!top) top = now;
      old = now,
      err = ReadLine(fp,buf,2048);
    }
  }
  NumberOfFunction = no,
  return((PLine)top);
}

// Make Registry Application Name
void MakeAppRegName(char *path,char *name,char *appregname)
{
  int i,len,

  strcpy(appregname,name),
  len = strlen(appregname);
  for( i = 0 , i < len , i++ ) {
    if(*(appregname + i) == '\\')
      *(appregname + i) = '/',
    if(*(appregname + i) >= 'A' && *(appregname + i) <= 'Z')
        *(appregname + i) = *(appregname + i) - 'A' + 'a';
  }
}

 id GetRealAPPName(char *appname)

 .nt i,len,flg;
 char buf[1024];
 flg = 0;
 len = strlen(appname);
 for(i = 0 , i < len , i++ ) {
   if(*(appname + i) == '"') {
     if(!flg)
       flg = i + 1,
     else
       *(appname + i) = '\0';
   }
 }
 strcpy(buf,appname + flg);
 strcpy(appname,buf);
}

void MakeNewFileName(char *file,char *newname)
{
  int i,len;
  len = strlen(file) + 1,
  for( i = len ; i > 0 ; i-- ) {
    if(*(file + i) == '\\' || *(file + i) == '/') {
      break;
    }
  }
  if(i != 0)
    strcpy(file + i + 1,newname),
  else
    strcpy(file,newname);
}

void MakeOneDynaTable(char *symbol_name);

    ****************************************************
   Dynamize· Make a Dynatab.dbj and change object (*.obj)
   file to  dynamized object file (* dbj).
   Dynamized Object file is the all of symbols
   which are defined in the object file, point
   to Dynatab.dbj

   Argments
   directory_of_object_file /a def_file_path_name
   directory_of_object_file /e
  ****************************************************/
static PLine def,

void main(int argc , char *argv[])
{
  char file[1024];
  char dbjf[1024];
  char buf[1024],
  char AppName[1024],
  WIN32_FIND_DATA dir,
  HANDLE data;
  int write,flg,
  unsigned long objd,dbjd;
```

```
  FILE *fp;
  PLine now;

  write = 1;
  flg = 0;

  if(argc < 2 || argc > 4) {
  Error·
    printf("Error dynaobj: wrong argments. Should give 2 or 3 argments  \n");
    printf("In case of an application, argments are direcory_of_object_files /a or /A def file pat
h name\n");
    printf("In case of an Empty dynatab.dbj, argments are directory /e or /E\n");
    exit(1);
  }

  if(strcmp(argv[2],"/a") == 0 || strcmp(argv[2],"/A") == 0) {
    /* Take Game APP name */
    DynaMaizedAPP();
    flg = 1;
  }
  else if (strcmp(argv[2],"/e") == 0 || strcmp(argv[2],"/E") == 0) {
    if(SetCurrentDirectory(argv[1]) == FALSE) {
      printf(" %s is not a directory\n",argv[1]);
      exit(1);
    }
    goto End;
  }
  else {
    goto Error;
  }

  OpenDebug();

  if(flg) { /* Application */
    /* Get  def File */
    if((fp = fopen(argv[3],"r")) == NULL) {
      printf("Can not Find %s \n",argv[3]);
#ifdef _DEBUG
      fprintf(debug_fp,"Can not Find %s \n",argv[3]);
#endif
      CloseDebug();
      exit(1),
    }
    /* Get full path def file */
    ReadLine(fp,buf,1024); // first line should be def file full path name
    strcpy(file,&buf[1]);
    ReadLine(fp,AppName,1024);

    GetRealAPPName(AppName),
    ReadLine(fp,buf,1024), /* Exports string ignore */
    def = LoadDefFileIntoMemory(fp); /* Read All function lines */
    fclose(fp),
    MakeNewFileName(file,"");
    MakeAppRegName(file,AppName,app_name);
  }

  if(SetCurrentDirectory(argv[1]) == FALSE) {
    printf(" %s is not a directory\n",argv[1]);
#ifdef _DEBUG
    fprintf(debug_fp,"%s is not a directory\n",argv[1]);
#endif
    CloseDebug(),
    exit(1),
  }

  if((data = FindFirstFile("* obj",&dir)) == INVALID_HANDLE_VALUE) {
    printf("Can not find object file in this directoy \n");
#ifdef _DEBUG
    fprintf(debug_fp,"Can not fid Object file in this direcory\n");
#endif
    CloseDebug(),
    exit(1),
  }

  for( , , ) {
    memset(file,0x00,1024),
```



```
    GetCurrentDirectory(1024,file);
    strcat(file,"\\");
    strcat(file,dir.cFileName);
    MakeDBJName(file,dbjf);
    objd = GetModifyTime(file);
    dbjd = GetModifyTime(dbjf),
    if(objd > dbjd) { /* Object file is new */
      /* Need Dinamized */
      //printf("Dynamized !! %s \n",file),
#ifdef _DEBUG
        fprintf(debug_fp,"File : %s \n",file);
#endif
      map_file(file);
      GetSymbolTableFromHeader(0),
      WriteDataToFile(dbjf),
    } else { /* Not Need Dinamized */
      map_file(dbjf);
      GetSymbolTableFromHeader(1);
    }
    init_contorl_table();
    if(FindNextFile(data,&dir) == FALSE)
      break;
  }
  /* Make Dynatable */
  for(now = def ; now ; now = now->next) {
    MakeOneDynaTable(now->fname);
  }
#if 0
  /* MFC Funciton DynaMize Test */
    MakeSymbolChainWithRelocation("__imp__FindResourceA@12","?KosakaFindResource@@YAPAUHRSRC__@@PA
UHINSTANCE__@@PBD1@Z");
#endif

End.
  GetCurrentDirectory(1024,file);
  strcat(file,"\\");
  strcat(file,"DynaTab.dbj");
  WriteDynaObject(file,flg);
  //    printf("DynaGenerate Done\n");

  CloseDebug();
  exit(0);
}

int FindInDef(char *fname)
{
    PLine now;
    if(fname[0] == '_') {
        for(now = def ; now ; now = now->next) {
            if(strcmp(now->fname,&fname[1]) == 0)
                return(1);
        }
    }
    else {
        for(now = def ; now ; now = now->next) {
            if(strcmp(now->fname,fname) == 0)
                return(1);
        }
    }
    return(0);
}
```

```
/*--------------------
  Common.c
  Takashi Kosaka 1996 SegaSoft Inc
  --------------------*/
#define WINDOWS
#ifdef WINDOWS
#include <windows.h>
#include <string.h>
#else
#include "winnt.h"
#endif

#include <stdio.h>

#define DYNAPLAY_SIG "DynaPlay(TM) by SegaSoft(C) T.K"
#define SIG_SIZE 31

typedef struct _reloc_chain {
  PIMAGE_RELOCATION reloc;
  struct _reloc_chain *next;
} RELOC_CHAIN, *PRELOC_CHAIN;

typedef struct _data_charin {
  unsigned char *data;
  int size,
  struct _data_charin *next;
} DATA_CHAIN, *PDATA_CHAIN,

typedef struct _symbol_chain {
  PIMAGE_SYMBOL symbol;
  struct _symbol_chain *next;
} SYMBOL_CHAIN, *PSYMBOL_CHAIN;

typedef struct _string_t_chain {
  char *name;
  struct _string_t_chain *next;
} STRING_T_CHAIN, *PSTRING_T_CHAIN,

typedef struct _symbol_name_chain {
  char *symbol_name;
  struct _symbol_name_chain *next,
} SYMBOL_NAME_CHAIN, *PSYMBOL_NAME_CHAIN;

#include "common.h"

static struct _OBJCONT {
  PRELOC_CHAIN toprel;
  PRELOC_CHAIN oldrel;
  PDATA_CHAIN topd;
  PDATA_CHAIN oldd,
  PSYMBOL_CHAIN topsym;
  PSYMBOL_CHAIN oldsym;
  PSTRING_T_CHAIN topst;
  PSTRING_T_CHAIN oldst;
  PSYMBOL_NAME_CHAIN topsymboln;
  PSYMBOL_NAME_CHAIN oldsymboln,
  int symbol_index;
  int no_of_reloc,
  int top_reloc_add,
  int top_symbol_add;
  DWORD next_data_add,
  int stringtable_size,
  int next_string_index;
} Object = {(PRELOC_CHAIN)NULL,(PRELOC_CHAIN)NULL,
           (PDATA_CHAIN)NULL,(PDATA_CHAIN)NULL,
           (PSYMBOL_CHAIN)NULL,(PSYMBOL_CHAIN)NULL,
           (PSTRING_T_CHAIN)NULL,(PSTRING_T_CHAIN)NULL,
           (PSYMBOL_NAME_CHAIN)NULL,(PSYMBOL_NAME_CHAIN)NULL,
           0,0,0,0,0,0,4);

void CreateDynaObject(int index,int flg),
static void SetStringTable(PIMAGE_SYMBOL pSymbolTable, int cSymbols),
static int GetSymbolTable(PIMAGE_SYMBOL pSymbolTable, int cSymbols,int flg),
void MakeStringChain(PSYMBOL_CHAIN now,char *symbol_name,int size) ;

char *stralloc(char *buf)
```

```
{
  char *new_buf;
  if((new_buf = (char *)malloc(strlen(buf) + 1)) == NULL) {
    return((char *)NULL);
  }
  strcpy(new_buf,buf);
  return((char *)new_buf);
}

/* Find source string from target string */
int search_string(char * source,char *target)
{
  int i,j,slen,tlen;
  slen = strlen(source),
  tlen = strlen(target);
  j = 0;
  for(i = 0 ; i < tlen ; i++ ) {
    if(*(target + i) == *(source + j)) {
      j++;
      if(j >= slen)
    return(1);
    }
    else
      j = 0;
  }
  return(0);
}

int map_file(char *file_name)
{
  FILE *fp;
  int i;
  int data,size;
  unsigned char *cdata;

  if((fp = fopen(file_name,"rb")) == NULL) {
    printf("Can not find %s file\n",file_name),
    exit(1);
  }

  fseek(fp,0L,2); /* Go to end of file */
  ocontrol size = size = ftell(fp);
  rewind(fp),

  /* memory get */
  if((ocontrol.pImageFileHeader = (PIMAGE_FILE_HEADER)malloc(size)) == NULL) {
    printf("Can not make a memory !!! \n"),
    exit(1);
  }
  cdata = (unsigned char *)ocontrol.pImageFileHeader;

  /* Memory Map */
  for( i = 0 , i < size ; i++) {
    data = fgetc(fp);

    *(cdata + i) = (unsigned char)data;
  }
  fclose(fp),
  if(strcmp((char *)(cdata + size - SIG_SIZE - 1),DYNAPLAY_SIG) == 0)
    return(1),
  else
    return(0),
}

static char * GetSectionNameFromSectionNo(int section)
{
  PIMAGE_SFCTION_HEADER pSections,
  pSections = (PIMAGE_SECTION_HEADER)(ocontrol.pImageFileHeader+1);
  return((char *)pSections[section -1] Name),
}

static PIMAGE_RELOCATION  GetRelocationDataFromSection(int section,
                               int *reloc_no)
{
  PIMAGE_SECTION_HEADER pSections,
  pSections = (PIMAGE_SECTION_HEADER)(ocontrol.pImageFileHeader+1);
  *reloc_no = pSections[section -1].NumberOfRelocations;
  return((PIMAGE_RELOCATION)MakePtr(PIMAGE_RELOCATION,
                    ocontrol pImageFileHeader,
                    pSections[section -1].PointerToRelocations));
}

/* If old_id is in relocation, return 1 othtewise 0 */
static int ReplaceSymbolIDInRelocation(DWORD old_id,DWORD new_id)
{
  int i,reloc_no,j,ret;
  PIMAGE_RELOCATION reloc;
  char *section_name;
  ret = 0,
  for(i = 1 , i <= ocontrol pImageFileHeader->NumberOfSections ; i++) {
    reloc = GetRelocationDataFromSection(i,&reloc_no);
    section_name = GetSectionNameFromSectionNo(i);
    if(!search_string(" debug",section_name)) {
      for(j = 0 , j < reloc_no ; j++ ) {
    if(reloc->SymbolTableIndex == old_id){
      reloc->SymbolTableIndex = new_id;
      ret = 1;
    }
    reloc++;
      }
    }
  }
  return(ret);
}

void MakeUndefineSymbolString(PSYMBOL_CHAIN now,char *symbol_name)
{
  int size;
  PSTRING_T_CHAIN snow;
  size = strlen(symbol_name) + 1;

  if(size > 7) {
    if((snow = (PSTRING_T_CHAIN)malloc(sizeof(STRING_T_CHAIN))) == NULL) {
      printf("Can not make memory for STRING_T_CHAIN \n");
      exit(1);
    }
    snow->next = (PSTRING_T_CHAIN)NULL;
    if(!ocontrol.topstring)  ocontrol.topstring = snow;
    if(ocontrol.oldstring) ocontrol.oldstring->next = snow,
    snow->name = stralloc(symbol_name);
    now->symbol->N.Name.Short = 0;
    now->symbol->N.Name Long = ocontrol.NextStringIndex;
    ocontrol.NextStringIndex += size;
    ocontrol.NewStringTableSize += size;
    ocontrol.oldstring = snow;
  }
  else {
    strcpy((char *)now->symbol->N.ShortName,symbol_name);
  }



void MakeUndefineSymbol(char *symbol_name,int symbol_id)
{
  PSYMBOL_CHAIN now,
  if(ReplaceSymbolIDInRelocation(symbol_id,ocontrol NextSymbolNumber)) {
    if((now = (PSYMBOL_CHAIN)malloc(sizeof(SYMBOL_CHAIN))) == NULL) {
      printf("Can not make memory for SYMBOL_CHAIN \n");
      exit(1);
    }

    if((now->symbol = (PIMAGE_SYMBOL)malloc(sizeof(IMAGE_SYMBOL)))
       == NULL) {
      printf("Can not make memory for IMAGE_SYMBOL \n"),
      exit(1);
    }
    now->next = (PSYMBOL_CHAIN)NULL;
    now->symbol->Value = 0x0000;
    now->symbol->SectionNumber = IMAGE_SYM_UNDEFINED,
    now->symbol->Type = 0x0020,
    now->symbol->StorageClass = IMAGE_SYM_CLASS_EXTERNAL;
    now->symbol->NumberOfAuxSymbols = 0;
```

```
    if(!ocontrol.topsymbol) ocontrol.topsymbol = now;
    if(ocontrol.oldsymbol) ocontrol.oldsymbol->next = now;
    ocontrol.oldsymbol = now;
    ocontrol.NextSymbolNumber++;
    MakeUndefineSymbolString(now,symbol_name),
  }
}
/* for DynaTab.obj */
PIMAGE_SYMBOL MakeNewDefineSymbol(char *symbol_name,long add,int section)
{
  PSYMBOL_CHAIN now,
  int size;
  if((now = (PSYMBOL_CHAIN)malloc(sizeof(SYMBOL_CHAIN))) == NULL) {
    printf("Can not make memory for SYMBOL_CHAIN \n");
    exit(1);
  }
  if((now->symbol = (PIMAGE_SYMBOL)malloc(sizeof(IMAGE_SYMBOL)))== NULL) {
    printf("Can not make memory for IMAGE_SYMBOL \n");
    exit(1);
  }
  now->next = (PSYMBOL_CHAIN)NULL;
  now->symbol->Value = add;
  now->symbol->SectionNumber = section;
  now->symbol->Type = 0x0020;
  now->symbol->StorageClass = IMAGE_SYM_CLASS_EXTERNAL;
  now->symbol->NumberOfAuxSymbols = 0;
  if(!Object topsym) Object.topsym = now;
  if(Object oldsym)  Object oldsym->next = now,
  Object.oldsym = now;
  Object symbol_index ++;
  size = strlen(symbol_name) + 1,
  MakeStringChain(now,symbol_name,size);
  return((PIMAGE_SYMBOL)now->symbol);
}

void WriteRowData(FILE *fp,unsigned char *data,int size)
{
  int i;
  for(i = 0 , i < size , i++ ) {
    putc(*data++,fp);
  }
}

void WriteDataToFile(char *file_name)
{
  FILE *fp;
  int i,data,
  unsigned char *tmp,null,
  PSYMBOL_CHAIN now,
  PSTRING_T_CHAIN snow;

  ocontrol.pImageFileHeader->NumberOfSymbols = ocontrol NextSymbolNumber;

  if((fp = fopen(file_name,"wb")) == NULL) {
    printf("Can not find %s file\n",file_name),
    exit(1);
  }
  tmp = (unsigned char *)ocontrol pImageFileHeader;

  /* Memory Map */
  for( i = 0 , i < ocontrol EndOfSymbolTable , i++) {
    data = (int)*tmp++,
    putc(data,fp),
  }

  /* Write NewSymbol Table */
  for(now = ocontrol topsymbol , now , now = now->next) {
    WriteRowData(fp,(unsigned char *)now->symbol,sizeof(IMAGE_SYMBOL)),
  }
  if(ocontrol.topsymbol || ocontrol topstring ) {
    //printf("New String Table Size: %d \n",ocontrol NewStringTableSize),
    WriteRowData(fp,(unsigned char *)&ocontrol.NewStringTableSize,4),
    i += 4,
    tmp += 4,
  }
  for( , i < ocontrol size ; i++ ) {
    data = (int)*tmp++;
    putc(data,fp);
  }
  for(snow = ocontrol.topstring , snow , snow = snow->next) {
    WriteRowData(fp,snow->name,strlen(snow->name) + 1);
  }
  /* DynaPlay Signiture */
  WriteRowData(fp,DYNAPLAY_SIG,SIG_SIZE);
  null = '\0';
  WriteRowData(fp,&null,1);
  fclose(fp);
}

static char * LookupSymbolName(DWORD index)
{
  static char shortname[9];

  if ( ocontrol.PCOFFSymbolTable[index].N Name Short != 0 ) {
    if(strlen((char *)ocontrol.PCOFFSymbolTable[index] N.ShortName) >= 8) {
      int i;
      for(i = 0 ; i < 8 , i++ ) {
    shortname[i] = *(ocontrol.PCOFFSymbolTable[index].N.ShortName + i);
      }
      shortname[i] = '\0';
      return(shortname);
    }
    else
      return((char *)ocontrol.PCOFFSymbolTable[index].N.ShortName),
  } else
    {
      return((char *)(ocontrol.stringTable +
              ocontrol.PCOFFSymbolTable[index].N.Name.Long));
    }
}

static char * SetDYNASymbolName(DWORD index)
{
  PIMAGE_SYMBOL symbol;
  PSTRING_T_CHAIN snow;
  char *now_symbol_name,*new_symbol_name;
  int len;

  symbol = (PIMAGE_SYMBOL)&ocontrol PCOFFSymbolTable[index];
  now_symbol_name = LookupSymbolName(index);

  if(now_symbol_name[0] == '_')
    len = strlen(now_symbol_name) + 5;
  else
    len = strlen(now_symbol_name) + 6,

  if((new_symbol_name = (char *)malloc(len)) == NULL) {
    printf("Can not Make a Memory for new Symbol Name\n"),
    exit(1);
  }
  strcpy(new_symbol_name,"?DYNA"),
  if(now_symbol_name[0] == '_')
    strcat(new_symbol_name,&now_symbol_name[1]);
  else
    strcat(new_symbol_name,now_symbol_name),


  symbol->N.Name.Short = 0;
  symbol->N.Name Long = ocontrol.NextStringIndex;
  ocontrol.NextStringIndex += len;
  ocontrol NewStringTableSize += len,

  if((snow = (PSTRING_T_CHAIN)malloc(sizeof(STRING_T_CHAIN))) == NULL) {
    printf("Can not Make Memory for STRING_T_CHAIN\n");
    exit(1);
  }
  snow->next = (PSTRING_T_CHAIN)NULL,
  if(!ocontrol.topstring) ocontrol.topstring = snow;
  if(ocontrol oldstring) ocontrol oldstring->next = snow;
  ocontrol oldstring = snow,
  snow->name = new_symbol_name,
  return((char *)new_symbol_name);
```

```
#ifdef OLD
  if ( ocontrol.PCOFFSymbolTable[index].N.Name.Short != 0 ) {
    if(strlen((char *)ocontrol.PCOFFSymbolTable[index].N.ShortName) >= 8) {
      for(i = 0 ; i < 8 , i++ )
    shortname[i] = *(ocontrol.PCOFFSymbolTable[index] N ShortName + i);
      shortname[i] = '\0';
    }
    else
      strcpy(shortname,ocontrol PCOFFSymbolTable[index].N ShortName),

    if(shortname[0] == '_')
      _strrev(shortname + 1);
    else
      _strrev(shortname);

    for(i = 0 ; i < 8 , i++ ) {
      *(ocontrol.PCOFFSymbolTable[index] N.ShortName + i) = shortname[i];
      if(!shortname[i])
    break;
    }
    return(shortname),
  }
  else  {
    if(*(ocontrol.stringTable + ocontrol.PCOFFSymbolTable[index].N.Name.Long) == '_')
      _strrev(ocontrol.stringTable + 1 +
          ocontrol.PCOFFSymbolTable[index].N.Name.Long);
    else
      _strrev(ocontrol stringTable +
          ocontrol.PCOFFSymbolTable[index].N.Name Long);

    return((char *)(ocontrol.stringTable +
            ocontrol PCOFFSymbolTable[index].N.Name.Long));
  }
#endif
}

static void ChangeSectionTableAttribute(WORD section)
{
  PIMAGE_SECTION_HEADER pSections;
  pSections = (PIMAGE_SECTION_HEADER)(ocontrol.pImageFileHeader+1);
  if(pSections[section -1] Characteristics  == 0x40400040)
    pSections[section -1].Characteristics |= IMAGE_SCN_MEM_WRITE,
}

static void SetStringTable(PIMAGE_SYMBOL pSymbolTable, int cSymbols)
{
  int size,tmp;
  /* The string table apparently starts right after the symbol table  */
  ocontrol.stringTable = (PSTR)&pSymbolTable[cSymbols];
  ocontrol EndOfSymbolTable = (int)ocontrol.stringTable - (int)ocontrol.pImageFileHeader,
  memcpy(&size,ocontrol.stringTable,4);
  tmp = ocontrol.size - size;
  //printf("End Symbol Table  %d tmp %d \n",ocontrol EndOfSymbolTable,tmp);
  ocontrol.NewStringTableSize = size,
  ocontrol NextStringIndex = size;
}

int FindInDef(char *fname);

static int GetSymbolTable(PIMAGE_SYMBOL pSymbolTable, int cSymbols,int flg)
{
  int i;
  char *symbol_name,
 // char * dynaplay_call_symbol,

  for ( i = 0; i < cSymbols; i++ ) {
    symbol_name = LookupSymbolName(i),
    if(pSymbolTable->SectionNumber > 0 &&
       pSymbolTable->StorageClass == IMAGE_SYM_CLASS_EXTERNAL &&
       ISFCN(pSymbolTable->Type)) {

      //if(dynaplay_call_symbol != symbol_name && strcmp(symbol_name,"_GetExportDynaTable")) {
            if(FindInDef(symbol_name))
                CreateDynaObject(i,flg);
      //}
  }
```

```
    /*  Take into account any aux symbols */
    i += pSymbolTable->NumberOfAuxSymbols;
    pSymbolTable += pSymbolTable->NumberOfAuxSymbols,
    pSymbolTable++,
  }
  return(0),
}

int GetSymbolTableFromHeader(int flg)
{
  int add;

  ocontrol.COFFSymbolCount = ocontrol.pImageFileHeader->NumberOfSymbols;

  add = ocontrol pImageFileHeader->PointerToSymbolTable;
  ocontrol PCOFFSymbolTable = MakePtr(PIMAGE_SYMBOL, ocontrol.pImageFileHeader,
                       add);
  ocontrol.NextSymbolNumber = ocontrol COFFSymbolCount,
  SetStringTable(ocontrol PCOFFSymbolTable,ocontrol.COFFSymbolCount),
  return(GetSymbolTable(ocontrol PCOFFSymbolTable,ocontrol.COFFSymbolCount,flg)),
}

static int IsDynaPlaySymbolBigAddr(int symbol_id,int section,DWORD Addr)
{
  int i,numberofreloc;
  PIMAGE_RELOCATION reloc;

  reloc = GetRelocationDataFromSection(section,&numberofreloc);
  for( i = 0 , i < numberofreloc ; i++ ) {
    if((int)reloc->SymbolTableIndex == symbol_id) {
      if(reloc->VirtualAddress > Addr)
    return(1);
    }
    reloc++,
  }
  return(0);
}


void MakeRowDataChian()
{
  PDATA_CHAIN now;
  int size = 8,

  if((now = (PDATA_CHAIN)malloc(sizeof(DATA_CHAIN))) == NULL) {
    printf("Can not make memory for PDATA_CHAIN \n");
    exit(1);
  }
  if((now->data = (unsigned char *)malloc(size)) == NULL) {
    printf("Can not make memory for RowData \n");
    exit(1);
  }

  now->next = (PDATA_CHAIN)NULL;
  now->size = size;
  if('Object topd) Object topd = now;
  if(Object oldd)  Object.oldd->next = now;
  /* *(now->data) = 0xE9;
   *(now->data + 1) = 0x00;
   *(now->data + 2) = 0x00;
   *(now->data + 3) = 0x00;
   *(now->data + 4) = 0x00,
   *(now->data + 5) = 0xFF,
   *(now->data + 6) = 0xFF;  */
  *(now->data)     = 0xb8;
  *(now->data + 1) = 0x00;
  *(now->data + 2) = 0x00,
  *(now->data + 3) = 0x00;
  *(now->data + 4) = 0x00,
  *(now->data + 5) = 0xFF,
  *(now->data + 6) = 0xE0;
  *(now->data + 7) = 0x00,
  Object.next_data_add += size,
  Object.top_symbol_add += size;
  Object.top_reloc_add += size,
  Object oldd = now;
```

```
}


void EncodeString(char *string)
{
  int len,i;
  unsigned char dd,ll,
  len = strlen(string);
  for(i = 0 ; i < len ; i++ ) {
    dd = (unsigned char)*(string + i),
    ll = 0x01 & dd;
    *(string + i) = dd >> 1 | dd << 7;
  }
}


void MakeRowDataChianByName(char *name)
{
  PDATA_CHAIN now;
  int size;
  size = strlen(name) + 1;

  if((now = (PDATA_CHAIN)malloc(sizeof(DATA_CHAIN))) == NULL) {
    printf("Can not make memory for PDATA_CHAIN \n");
    exit(1);
  }
  if((now->data = (unsigned char *)malloc(size)) == NULL) {
    printf("Can not make memory for RowData \n");
    exit(1);
  }
  now->size = size;
  now->next = (PDATA_CHAIN)NULL,
  if('Object topd) Object.topd = now,
  if(Object.oldd)  Object oldd->next = now,
  strcpy(now->data,name);
  EncodeString(now->data);
  Object.next_data_add += size;
  Object.top_symbol_add += size;
  Object.top_reloc_add += size;
  Object oldd = now;
}

PRELOC_CHAIN MakeRelocationChian()
{
  PRELOC_CHAIN now;

  if((now = (PRELOC_CHAIN)malloc(sizeof(RELOC_CHAIN))) == NULL) {
    printf("Can not make memory for RELOC_CHAIN \n");
    exit(1);
  }
  if((now->reloc = (PIMAGE_RELOCATION)malloc(sizeof(IMAGE_RELOCATION)))
     == NULL) {
    printf("Can not make memory for RELOCATION \n");
    exit(1),
  }
  now->next = (PRELOC_CHAIN)NULL;
  if('Object toprel) Object toprel = now,
  if(Object.oldrel)  Object oldrel->next = now;
  Object no_of_reloc ++;
  Object top_symbol_add += sizeof(IMAGE_RELOCATION);
  Object.oldrel = now,
  return((PRELOC_CHAIN)now),
}

void MakeStringChain(PSYMBOL_CHAIN now,char *symbol_name,int size)
{
  PSTRING_T_CHAIN snow;
  if(size > 7) {
    if((snow = (PSTRING_T_CHAIN)malloc(sizeof(STRING_T_CHAIN))) == NULL) {
      printf("Can not make memory for STRING_T_CHAIN \n");
      exit(1);
    }
    snow->next = (PSTRING_T_CHAIN)NULL;
    if('Object topst) Object topst = snow,
    if(Object.oldst) Object oldst->next = snow;
    snow->name = stralloc(symbol_name);
    now->symbol->N.Name.Short = 0;
    now->symbol->N Name.Long = Object.next_string_index;
    Object next_string_index += size;
    Object.stringtable_size += size;
    Object.oldst = snow;
  }
  else {
    strcpy((char *)now->symbol->N.ShortName,symbol_name);
  }
}


void MakeSymbolChainWithRelocation(char * symbol_name,char *new_name)
{
  PSYMBOL_CHAIN now;
  PRELOC_CHAIN lnow;
  int size,now_row_add,

   f((now = (PSYMBOL_CHAIN)malloc(sizeof(SYMBOL_CHAIN))) == NULL) {
    printf("Can not make memory for SYMBOL_CHAIN \n");
    exit(1);
  }
  if((now->symbol = (PIMAGE_SYMBOL)malloc(sizeof(IMAGE_SYMBOL)))
      == NULL) {
    printf("Can not make memory for IMAGE_SYMBOL \n");
    exit(1),
  }
  now->next = (PSYMBOL_CHAIN)NULL;
  if(!Object.topsym) Object topsym = now;
  if(Object.oldsym)  Object.oldsym->next = now;
  Object.oldsym = now;

  size = strlen(symbol_name) + 1;
  now_row_add = now->symbol->Value = Object.next_data_add;
  now->symbol->SectionNumber = 1;
  now->symbol->Type = 0x0000;
  now->symbol->StorageClass = IMAGE_SYM_CLASS_EXTERNAL;
  now->symbol->NumberOfAuxSymbols = 0;
  MakeRowDataChian(),
  MakeStringChain(now,symbol_name,size);
  //WriteSymbolName(symbol_name);
  /* Not need Symbol Name for DynaTable */
  //MakeRowDataChianByName(symbol_name);

   if((now->next = (PSYMBOL_CHAIN)malloc(sizeof(SYMBOL_CHAIN))) == NULL) {
     printf("Can not make memory for SYMBOL_CHAIN \n");
     exit(1);
   }

   if((now->next->symbol = (PIMAGE_SYMBOL)malloc(sizeof(IMAGE_SYMBOL)))
       == NULL) {
     printf("Can not make memory for IMAGE_SYMBOL \n");
     exit(1);

   now->next->next = (PSYMBOL_CHAIN)NULL;
   Object oldsym = now->next,
   now->next->symbol->Value = 0x0000;
   now->next->symbol->SectionNumber = IMAGE_SYM_UNDEFINED;
   now->next->symbol->Type = 0x0020,
   now->next->symbol->StorageClass = IMAGE_SYM_CLASS_EXTERNAL,
   now->next->symbol->NumberOfAuxSymbols = 0;

   /* relocation table */
   lnow = MakeRelocationChian(),
   lnow->reloc->VirtualAddress = now_row_add + 1;
   lnow->reloc->SymbolTableIndex = Object symbol_index + 1,
   lnow->reloc->Type = IMAGE_REL_I386_DIR32;
   size = strlen(new_name) + 1,
   MakeStringChain(now->next,new_name,size);

   Object.symbol_index += 2;

   return;
 }

 /* Make Dynatable in Dynatab dbj */
```

```
void MakeOneDynaTable(char *symbol_name)
{
  char *new_name,
  char *org_name,
  int len;
  len = strlen(symbol_name) + 6,

  if((new_name = (char *)malloc(len)) == NULL) {
    printf("Can not Make new symbol name Memory \n");
    exit(1);
  }
  if(symbol_name[0] != '?') {
      if((org_name = (char *)malloc(strlen(symbol_name) + 2)) == NULL) {
        printf("Can not Make new symbol name Memory \n");
         exit(1);
      }
      strcpy(org_name,"_");
      strcat(org_name,symbol_name);
  }
  else
      org_name = symbol_name;

  strcpy(new_name,"?DYNA"),
  strcat(new_name,symbol_name);
  MakeSymbolChainWithRelocation(org_name,new_name);
}

void MakeSymbolChain(char * symbol_name)
{
  PSYMBOL_CHAIN now;
  int size;

  if((now = (PSYMBOL_CHAIN)malloc(sizeof(SYMBOL_CHAIN))) == NULL) {
    printf("Can not make memory for SYMBOL_CHAIN \n");
    exit(1);
  }
  if((now->symbol = (PIMAGE_SYMBOL)malloc(sizeof(IMAGE_SYMBOL)))
      == NULL) {
    printf("Can not make memory for IMAGE_SYMBOL \n");
    exit(1);
  }
  now->next = (PSYMBOL_CHAIN)NULL,
  if(!Object.topsym) Object.topsym = now;
  if(Object oldsym)  Object.oldsym->next = now;
  Object.oldsym = now;

  size = strlen(symbol_name) + 1;
  now->symbol->Value = Object.next_data_add,
  now->symbol->SectionNumber = 1;
  now->symbol->Type = 0x0000;
  now->symbol->StorageClass = IMAGE_SYM_CLASS_EXTERNAL;
  now->symbol->NumberOfAuxSymbols = 0;
  MakeStringChain(now,symbol_name,size);
  Object symbol_index++,
  MakeRowDataChianByName("        "),

  return,
}

/* This function Must call at first */
void DynaMaizedAPP()
{
  Object top_reloc_add =
    sizeof(IMAGE_FILE_HEADER) + IMAGE_SIZEOF_SECTION_HEADER;
  Object top_symbol_add = Object.top_reloc_add;

  MakeSymbolChain("_DynaMizedAPPMain"),
}

static int FindSameSymbol(char *symbol_name)
{
  PSYMBOL_NAME_CHAIN now,
  for(now = Object.topsymboln , now ; now = now->next) {
    if(strcmp(now->symbol_name,symbol_name) == 0)
      return(1);
  }
  return(0);
}

/* This function Must call at first */
void DynaModAPP()
{
  Object top_reloc_add =
    sizeof(IMAGE_FILE_HEADER) + IMAGE_SIZEOF_SECTION_HEADER;
  Object top_symbol_add = Object.top_reloc_add;

  MakeSymbolChain("_DynaModAPPMain");
}

char * RealSymbolName(char *symbol_name)
{
  return(symbol_name + 5);

  /*
    if(*symbol_name == '_')
    _strrev(symbol_name + 1);
    else
    _strrev(symbol_name);
    */
}

void CreateDynaObject(int index,int flg)
{
  PSYMBOL_NAME_CHAIN now,
  char *org_name,*new_name;

  if(!flg) { /* Not Dynamized file */
    org_name = stralloc(LookupSymbolName(index));

    if(!FindSameSymbol(org_name)) {
      if((now = (PSYMBOL_NAME_CHAIN)malloc(sizeof(SYMBOL_NAME_CHAIN))) == NULL) {
    printf("Can not Make a memory for PSYMBOL_NAME_CHAIN\n");
    exit(1),
      }
      now->symbol_name = org_name,
      now->next = (PSYMBOL_NAME_CHAIN)NULL;
      if(!Object.topsymboln) Object topsymboln = now;
      if(Object.oldsymboln) Object.oldsymboln->next = now;
      Object oldsymboln = now;

      new_name = SetDYNASymbolName(index),
      //printf("DynaTable Symbol: %s \n",new_name);
      //MakeSymbolChainWithRelocation(org_name,new_name);
      MakeUndefineSymbol(org_name,index);
    }
    else {
      SetDYNASymbolName(index);
    }
  } else { /* Already DynaMized file */
    org_name = stralloc(RealSymbolName(LookupSymbolName(index)));  /* Get Undynamized name */
    if(!FindSameSymbol(org_name)) {
      if((now = (PSYMBOL_NAME_CHAIN)malloc(sizeof(SYMBOL_NAME_CHAIN))) == NULL) {
    printf("Can not Make a memory for PSYMBOL_NAME_CHAIN\n"),
    exit(1),
      }
      now->symbol_name = org_name,
      now->next = (PSYMBOL_NAME_CHAIN)NULL,
      if(!Object.topsymboln) Object topsymboln = now,
      if(Object oldsymboln) Object.oldsymboln->next = now,
      Object oldsymboln = now;

      new_name = LookupSymbolName(index);
      //MakeSymbolChainWithRelocation(org_name,new_name),
    }
  }
}

void my_memcpy(unsigned char *data,int atai)
{
  *data = 0xFF000000 & atai,
  *(data + 1) = 0x00FF0000 & atai;
  *(data + 2) = 0x0000FF00 & atai,
```

```
  *(data + 3) = 0x000000FF & atai;
}

/* Only One time Called */
IMAGE_SECTION_HEADER crt,
IMAGE_RELOCATION crt_reloc = {0,0,IMAGE_REL_I386_DIR32};
unsigned char crt_row[] = {0x00,0x00,0x00,0x00};


IMAGE_SECTION_HEADER text;


unsigned char text_row[] = {0x68, 0x00, 0x00, 0x00, 0x00,
                            0x68, 0x00, 0x00, 0x00, 0x00,
                            0x68, 0x00, 0x00, 0x00, 0x00,
                            0xb8, 0x00, 0x00, 0x00, 0x00,
                            0xff, 0xd0, 0x83, 0xc4, 0x0c,
                            0xc3, 0x90, 0x90, 0x90, 0x90,
                            0x90, 0x90};


#define app_imgae_size 32


/* IMAGE_RELOCATION text_reloc_dynamized = {0x00000007,0,IMAGE_REL_I386_DIR32},
   IMAGE_RELOCATION text_reloc_init      = {0x0000000c,0x00000000,IMAGE_REL_I386_DIR32},
   IMAGE_RELOCATION text_reloc_dynamain  = {0x00000011,0x00000000,IMAGE_REL_I386_DIR32},
   */

/* Relocation Settting *                    Address text*/
IMAGE_RELOCATION text_reloc_dynamized = {0x00000001,0,IMAGE_REL_I386_DIR32};
IMAGE_RELOCATION text_reloc_app_name  = {0x00000006,0,IMAGE_REL_I386_DIR32};
IMAGE_RELOCATION text_reloc_init      = {0x0000000B,0,IMAGE_REL_I386_DIR32};
IMAGE_RELOCATION text_reloc_dynamain  = {0x00000010,0,IMAGE_REL_I386_DIR32};


IMAGE_SECTION_HEADER dataa,
unsigned char *data_row,
int sizeofdata_row;
int AddSectionSize = 0,
extern char app_name[256];

void CreateDynaTableCallSection()
{
  int row_add;
  PIMAGE_SYMBOL symbol,
  /* IMAGE_AUX_SYMBOL aux,  */

  row_add = Object top_symbol_add   + sizeof(IMAGE_SECTION_HEADER) * 3,

  strcpy(crt Name," CRT$XCU"),
  crt Misc PhysicalAddress = 0;
  crt VirtualAddress = 0;
  crt SizeOfRawData = 4;
  crt PointerToRawData = row_add,
  crt PointerToRelocations = row_add + 4;
  crt.NumberOfRelocations = 1,
  crt NumberOfLinenumbers = 0,
  crt.Characteristics = 0xC0300040,
  crt_reloc SymbolTableIndex = Object symbol_index,
  Object top_symbol_add += sizeof(IMAGE_SECTION_HEADER) + sizeof(IMAGE_RELOCATION) + 4,
  symbol = MakeNewDefineSymbol("?$E$$$$$$190",0,3);
  symbol->StorageClass = IMAGE_SYM_CLASS_STATIC,

  row_add += sizeof(IMAGE_RELOCATION) + 4;

  //strcpy(text Name," rdata");
  strcpy(text.Name," text");
  text Misc.PhysicalAddress = 0,
  text.VirtualAddress = 0;
  text SizeOfRawData = app_imgae_size,
  text.PointerToRawData = row_add,
  text PointerToRelocations = row_add + app_imgae_size,

  text NumberOfRelocations = 4,
 text.NumberOfLinenumbers = 0,
 text.Characteristics = 0x60500020;

 text_reloc_app_name.SymbolTableIndex = Object.symbol_index,
 symbol = MakeNewDefineSymbol("$$AplicationName",9,4);
 symbol->Type = 0; /* Application name data */

 text_reloc_init SymbolTableIndex = Object.symbol_index;
 symbol = MakeNewDefineSymbol("$$DynaInit",0,4);
 symbol->Type = 0;   /* Name of init.dat Data */

 text_reloc_dynamain.SymbolTableIndex = Object.symbol_index;
 symbol = MakeNewDefineSymbol("_dynaplay_main",0,0);
 symbol->SectionNumber = IMAGE_SYM_UNDEFINED;

 Object.top_symbol_add += (sizeof(IMAGE_SECTION_HEADER) + sizeof(IMAGE_RELOCATION) * 4 + app_imga
e_size);
 row_add += sizeof(IMAGE_RELOCATION) * 4 + app_imgae_size;

 sizeofdata_row= 10 + strlen(app_name);

 strcpy(dataa.Name,".data");
 dataa.Misc PhysicalAddress = 0;
 dataa.VirtualAddress = 0;
 dataa.SizeOfRawData = sizeofdata_row;
 dataa.PointerToRawData = row_add;
 dataa.PointerToRelocations = 0;
 dataa.NumberOfRelocations = 0;
 dataa.NumberOfLinenumbers = 0,
 dataa.Characteristics = 0xC0400040;
 Object.top_symbol_add += (sizeof(IMAGE_SECTION_HEADER) + sizeofdata_row);
 AddSectionSize += sizeof(IMAGE_SECTION_HEADER) * 3;
 if((data_row = malloc(sizeofdata_row)) == NULL) {
   printf("Can Not Memory Allocation in RawData\n");
   exit(1);
 }
 strcpy(data_row,"init dat"),
 strcpy(data_row + 9,app_name);
}

void WriteDynaObject(char *name,int flg)
{
  FILE *fp;
  int end,
  int file_p;
  IMAGE_FILE_HEADER header,
  IMAGE_SECTION_HEADER sectionheadr;
  PDATA_CHAIN data;
  PRELOC_CHAIN reloc;
  PSYMBOL_CHAIN symbol;
  PSTRING_T_CHAIN string;

  if(flg == 1)
    CreateDynaTableCallSection();

  if((fp = fopen(name,"wb")) == NULL) {
    printf("Can not Create %s file\n",name);
    exit(1);
  }
  header Machine = 0x014c;
  if(flg == 0)
    header.NumberOfSections = 0,
  else if( flg == 1 )
    header NumberOfSections = 4;

  header.TimeDateStamp = 0x32fa92a2,
  header PointerToSymbolTable = Object top_symbol_add,
  header.PointerToSymbolTable = Object.top_symbol_add,
  header.NumberOfSymbols = Object.symbol_index;

  header.SizeOfOptionalHeader = 0;
  header Characteristics = 0,

  WriteRowData(fp,(unsigned char *)&header,sizeof(IMAGE_FILE_HEADER));
```

```
if(flg) {
  strcpy(sectionheadr.Name,".data");
  sectionheadr Misc PhysicalAddress = 0;
  sectionheadr VirtualAddress = 0;
  sectionheadr.SizeOfRawData = Object.next_data_add,
  sectionheadr.PointerToRawData =
    sizeof(IMAGE_FILE_HEADER) + IMAGE_SIZEOF_SECTION_HEADER + AddSectionSize;
  sectionheadr PointerToRelocations = Object.top_reloc_add + AddSectionSize;
  sectionheadr.PointerToLinenumbers = 0,
  sectionheadr NumberOfRelocations = Object.no_of_reloc;
  sectionheadr.NumberOfLinenumbers = 0;
  sectionheadr.Characteristics = 0xC0400040;

  WriteRowData(fp,(unsigned char *)&sectionheadr,IMAGE_SIZEOF_SECTION_HEADER);
  if(flg == 1) {
    WriteRowData(fp,(unsigned char *)&crt,IMAGE_SIZEOF_SECTION_HEADER);
    WriteRowData(fp,(unsigned char *)&text,IMAGE_SIZEOF_SECTION_HEADER),
    WriteRowData(fp,(unsigned char *)&dataa,IMAGE_SIZEOF_SECTION_HEADER);
  }
  /* printf("Set relocation Offset: %x\nSizeof RowData:%x\nSymbol Table Offset· %x\n",
     sectionheadr.PointerToRelocations,
     sectionheadr.SizeOfRawData,
     header.PointerToSymbolTable);
     */
  file_p = ftell(fp);
  //printf("Real Start RowData:%x \n",file_p);

  /* Row Data */
  for(data = Object topd ; data ; data = data->next) {
    if(data == Object.topd) {
  my_memcpy(data->data + 1,Object.no_of_reloc);
    }
    WriteRowData(fp,(unsigned char *)data->data,data->size);
  }

  //printf("Real Sizeof RowData.%x Start Relocation:%x \n",ftell(fp) - file_p,ftell(fp));

  /* Relocation Table */
  for(reloc = Object toprel ; reloc ; reloc = reloc->next) {
    WriteRowData(fp,(unsigned char *)reloc->reloc,sizeof(IMAGE_RELOCATION));
  }
  //printf("Real Start Symbol Table:%x \n",ftell(fp));

  if(flg == 1 ) {
    WriteRowData(fp,(unsigned char *)crt_row,4);
    WriteRowData(fp,(unsigned char *)&crt_reloc,sizeof(IMAGE_RELOCATION));

    WriteRowData(fp,(unsigned char *)text_row,app_imgae_size);

    WriteRowData(fp,(unsigned char *)&text_reloc_dynamized,sizeof(IMAGE_RELOCATION));
    WriteRowData(fp,(unsigned char *)&text_reloc_app_name,sizeof(IMAGE_RELOCATION));
    WriteRowData(fp,(unsigned char *)&text_reloc_init,sizeof(IMAGE_RELOCATION));
    WriteRowData(fp,(unsigned char *)&text_reloc_dynamain,sizeof(IMAGE_RELOCATION)),
    WriteRowData(fp,(unsigned char *)data_row,sizeofdata_row);
  }
  /* Symbol Table */
  for(symbol = Object.topsym ; symbol ; symbol = symbol->next) {
    WriteRowData(fp,(unsigned char *)symbol->symbol,sizeof(IMAGE_SYMBOL)),
  }
  Object.stringtable_size += 4,
  WriteRowData(fp,(unsigned char *)&Object stringtable_size,4);
  //printf("Size of String Table: %d \n",Object.stringtable_size);

  end = 0,
  /* String Table */
  for( string = Object.topst ; string , string = string->next) {
    WriteRowData(fp,(unsigned char *)string->name,strlen(string->name) + 1);
    end += strlen(string->name) + 1;
  }
  //printf("Real String Table Size: %d \n",end);
  end = 0x00;
 }
 //WriteRowData(fp,(unsigned char *)&end,1);

 fclose(fp);
}
```

DynaLib.DLL

Dynlib_main.c (main pert of Dynaplay)

```
/***************************************************************
  dynlib_main.c  (dynalib main function)
      By Takashi Kosaka (C) SegaSoft INC.    1997 -

  SEGASOFT CONFIDENTIAL - Unpublished Copyright (c) [1997].
  SegaSoft, Inc.  All Rights Reserved.
****************************************************************/

#include "scheme.h"
#ifdef FILES_HAVE_FDS
#include <sys/types h>
#include <sys/time.h>
#ifdef SELECT_INCLUDE
#include <sys/select.h>
#endif
#endif
#ifdef UNISTD_INCLUDE
#include <unistd.h>
   ndif
   def MACINTOSH_EVENTS
   nclude <Events.h>
#endif
#ifdef MACINTOSH_SIOUX
#include <console.h>
#include <SIOUX h>
#endif
#ifdef MACINTOSH_SET_STACK
#include <Memory.h>
#endif
#ifdef MACINTOSH_EVENTS
#include "simpledrop.h"
#endif
#ifdef UNIX_DYNAMIC_LOAD
#include <dlfcn.h>
#endif
#ifdef AIX_DYNAMIC_LOAD
#include  "../aixdlfcn/dlfcn h"
#define UNIX_DYNAMIC_LOAD
#endif
#ifdef WIN32
#include <direct h>
#include <stdlib.h>
#include <stdio h>
#include <sys/types h>
#include <sys/stat h>
#include <time.h>
#include <errno h>
#include <windows.h>
#define DllExport    __declspec( dllexport )
#endif

/* CFS Interface */
  nclude "cfs.h"


static Scheme_Env *global_env = (Scheme_Env *)NULL;

/* These strucures for the value from APP to Scheme */
typedef struct dynaplay_app_value (
  char *value_pointer,
  int length;
)DYNA_VALUE;

typedef struct list_dynaplay_app_value (
  char *value_pointer;
  int length;
  char value_name[32];
  struct list_dynaplay_app_value *next,
)LIST_DYNA_VALUE,*PLIST_DYNA_VALUE,

static PLIST_DYNA_VALUE top_value_list = (PLIST_DYNA_VALUE)NULL;
static PLIST_DYNA_VALUE old_value_list = (PLIST_DYNA_VALUE)NULL;

/* Function Table */
typedef struct _FATABLE (
  unsigned char top;
```

```
  union (
    unsigned int add;
    unsigned char tadd[4]:
  )N,
  char buf[3];
) FTABLE,

typedef struct _dll_table (
  int index;
  unsigned char add[4],
) DLL_TABLE, *PDLL_TABLE;

// Resource Swapping Data structure
typedef struct _ResourceSwap (
    unsigned long int Original;
    int Size;
    unsigned long int Point;
    struct _ResourceSwap * next;
) ResourceSwap , *PResourceSwap;

/* Open DLL control strucutre */
typedef struct _dll_handle (
  char *path;
#ifdef UNIX_DYNAMIC_LOAD
  void *dll;
#else
  HINSTANCE dll;
#endif
  int SizeOfDllTable;
  PDLL_TABLE *dll_tablé;
  void (*disable_func)();
  struct _dll_handle *next;
  PResourceSwap Pointer;
) DLL_HANDLE, *PDLL_HANDLE;

/* Application control handle */
typedef struct _app_handle (
    unsigned char *app_table;
    char *app_name;
    struct _app_handle *next,
    int vfs_mount_flg;          // VFS mount flag 1· mounted, other wize not mounted
    char *mount_path,           // VFS mount path name;
) APP_HANDLE, *PAPP_HANDLE;

static PDLL_HANDLE top_dll_handle = (PDLL_HANDLE)NULL;
static PDLL_HANDLE old_dll_handle = (PDLL_HANDLE)NULL;
static PAPP_HANDLE top_app_handle = (PAPP_HANDLE)NULL;
static PAPP_HANDLE old_app_handle = (PAPP_HANDLE)NULL;

/* Current Work Spave Directory */
static char CurrentWksDir[1024];

/* Export Functions for Windows */
#ifdef WIN32
DllExport int CFSMountWithPath(char *app_name),
DllExport int CreateCFS(char *path_name);
DllExport int dynaplay_main(char *file_name,char *app_name,unsigned char *table),
DllExport void init_scheme(),
DllExport unsigned long DynaEvalString(const char *eval_body,int *type),
DllExport int LoadNewScript(char *file_name);
#endif

/*****************************************************************
  Dynalib defines  Scheme Functions
*****************************************************************/
static Scheme_Object *app_value_string(int argc, Scheme_Object **argv),
static Scheme_Object *enable_dynamod(int argc, Scheme_Object **argv),
static Scheme_Object *disable_dynamod(int argc, Scheme_Object **argv);
static Scheme_Object *mount_cfs(int argc, Scheme_Object **argv);
static Scheme_Object *create_cfs(int argc, Scheme_Object **argv),
static Scheme_Object *umount_cfs(int argc, Scheme_Object **argv),
static Scheme_Object *create_registry(int argc, Scheme_Object **argv),
static Scheme_Object *use_registry(int argc, Scheme_Object **argv),
static Scheme_Object *delete_registry(int argc, Scheme_Object **argv);
static Scheme_Object *set_app_value(int argc, Scheme_Object **argv);
```

```
/* CFS (vertual file system interface) */
static int CheckCFSandRegistry(unsigned char *data);
static int CFSMount(char *app_name, PAPP_HANDLE now);
int CreateCFS(char *path_name);
int HardCPCreateCFS(char *path_name);
int CreateRegistry(char *app_name);
unsigned long DynaEvalString(const char *eval_body, int *type),
int SetRegistryValue(unsigned char *data, char *app_name);
int GetRegistryValue(unsigned char *data,char *app_name);
static int CheckCFS(char *app_name);
static int DeleteRegistry(char *app_name);

/****************************************************
    Exculsive Operation for Any Threads
*****************************************************/
#ifdef USE_WIN32_THREADS
extern HANDLE GC_allocate_ml,
   These Macros are LOCK/UNLOCK Process

     define LOCK(X) \
    static HANDLE now##X = (HANDLE)NULL; \
    if(!now##X) \
        now##X = CreateMutex(0,FALSE,"Lock"); \
    WaitForSingleObject(now##X, INFINITE); \

#    define UNLOCK(X) ReleaseMutex(now##X)

#endif

/*****************************************************
          debug Routine
******************************************************/
#define DynaPrintf(X,Y) dynaprintf(X,(LPVOID *)Y)
void dynaprintf(char *format , LPVOID data )
{
#ifdef _DEBUG
    FILE *fp,
    time_t ltime;
    struct tm *tm;

    time( &ltime );
    tm = localtime( &ltime );

    fp = fopen("DynaDebug.txt","a+");
    fprintf(fp,"%d:%d:%d ",tm->tm_hour,tm->tm_min,tm->tm_sec);
    fprintf(fp,format,data);
    fclose(fp);
#endif
}

void DynaDebugInit()

  fdef _DEBUG
    FILE *fp;
    time_t ltime,
    time( &ltime );
    fp = fopen("DynaDebug.txt","w");
    fprintf(fp,"DynaPlay Debug %s \n",ctime( &ltime ));
    fclose(fp);
#endif
}

void DynaDebug(char *buf)
{
#ifdef _DEBUG
    time_t ltime,
    struct tm *tm;
    FILE *fp;

    time( &ltime );
    tm = localtime( &ltime );

    fp = fopen("DynaDebug txt","a+");
    fprintf(fp,"%d:%d:%d ",tm->tm_hour,tm->tm_min,tm->tm_sec);
    fprintf(fp,buf);
    fclose(fp),
```

```
#endif
}
/*************************** End of Debug Rountine ****************/

#define CharTo4Bytes1(src,dst,off) \
        *(dst + off)     |= (0x81 & src); \
        *(dst + 1 + off) |= (0x12 & src); \
        *(dst + 2 + off) |= (0x28 & src); \
        *(dst + 3 + off) |= (0x44 & src)

#define CharFrom4Bytes1(dst,off) \
        (*(dst + off)     & 0x81) | \
        (*(dst + 1 + off) & 0x12) | \
        (*(dst + 2 + off) & 0x28) | \
        (*(dst + 3 + off) & 0x44)

#define CharTo4Bytes2(src,dst,off) \
        *(dst + off)     |= (0x14 & src); \
        *(dst + 1 + off) |= (0x28 & src); \
        *(dst + 2 + off) |= (0x42 & src); \
        *(dst + 3 + off) |= (0x81 & src)

#define CharFrom4Bytes2(dst,off) \
        ((*(dst + off)     & 0x14) | \
         (*(dst + 1 + off) & 0x28) | \
         (*(dst + 2 + off) & 0x42) | \
         (*(dst + 3 + off) & 0x81) )

#define IntTo16Bytes(src,dst,off) \
        CharTo4Bytes2((0xFF000000 & src) >> 24,dst,off     ); \
        CharTo4Bytes2((0x00FF0000 & src) >> 16,dst,off + 4 ); \
        CharTo4Bytes2((0x0000FF00 & src) >> 8, dst,off + 8 ); \
        CharTo4Bytes2((0x000000FF & src),      dst,off + 12)

#define IntFrom16Bytes(dst,off) \
        (((CharFrom4Bytes2(dst,off)    << 24)) | \
         ((CharFrom4Bytes2(dst,off + 4) << 16)) | \
         ((CharFrom4Bytes2(dst,off + 8)  << 8)) | \
         (CharFrom4Bytes2(dst,off + 12)))

/* src is Harf Size of Char */
unsigned char SetPattanChar(src,pat)
     unsigned char src,pat,
{
  unsigned char ret,mask,mm;
  int i;
  mm = mask = 1;
  ret = pat;
  for(i = 0 ; i < 8 ; i++ ) {
    if(mask & pat) {
      if(!(mm & src))
    ret ^= mask;
      mm <<= 1,
    }
    mask <<= 1;
  }
  return(ret),
}

/* src is a Full Size of Char */
unsigned char GetPattanChar(src,pat)
     unsigned char src,pat;
{
  unsigned char ret,mask,mm;
  int i;
  mm = mask = 1;
  ret = 0,
  src &= pat,
  for(i = 0 , i < 8 , i++ ) {
    if(mask & pat) {
      if(mask & src)
    ret |= mm,
      mm <<= 1;
    }
    mask <<= 1;
  }
  return(ret);
}

#define IntTo8Bytes(src,dst,off) \
  *(dst + off)     &= -0x6a; \
  *(dst + 1 + off) &= -0xC5; \
  *(dst + 2 + off) &= -0x95; \
  *(dst + 3 + off) &= ~0x3A; \
  *(dst + 4 + off) &= ~0x6A; \
  *(dst + 5 + off) &= -0xC5; \
  *(dst + 6 + off) &= -0x95; \
  *(dst + 7 + off) &= -0x3A; \
  *(dst + off)     |= SetPattanChar((0xF0000000 & src)>>28,0x6a); \
  *(dst + 1 + off) |= SetPattanChar((0x0F000000 & src)>>24,0xC5); \
  *(dst + 2 + off) |= SetPattanChar((0x00F00000 & src)>>20,0x95); \
  *(dst + 3 + off) |= SetPattanChar((0x000F0000 & src)>>16,0x3A); \
  *(dst + 4 + off) |= SetPattanChar((0x0000F000 & src)>>12,0x6A); \
  *(dst + 5 + off) |= SetPattanChar((0x00000F00 & src)>> 8,0xC5); \
   (dst + 6 + off) |= SetPattanChar((0x000000F0 & src)>> 4,0x95); \
  *(dst + 7 + off) |= SetPattanChar( 0x0000000F & src ,    0x3A)

#define IntFrom8Bytes(dst,off) \
  GetPattanChar(*(dst + off     ) , 0x6A) << 28  | \
  GetPattanChar(*(dst + 1 + off) , 0xC5) << 24  | \
  GetPattanChar(*(dst + 2 + off) , 0x95) << 20  | \
  GetPattanChar(*(dst + 3 + off) , 0x3A) << 16  | \
  GetPattanChar(*(dst + 4 + off) , 0x6A) << 12  | \
  GetPattanChar(*(dst + 5 + off) , 0xC5) << 8   | \
  GetPattanChar(*(dst + 6 + off) , 0x95) << 4   | \
  GetPattanChar(*(dst + 7 + off) , 0x3A)

static void RandomValueSet(unsigned char *dst,unsigned long dd);

/*  Initialize Current Wks Dir
    Current Wks Dir is the same directory where application exists */
static void InitCurrentWksDir(char *app_path)
{
    int i,len;

    len = strlen(app_path);

    strcpy(CurrentWksDir,app_path);

    for(i = len ; i >= 0 ; i-- ) {
        if(*(app_path + i) == '/' || *(app_path + i) == '\\' ||
            *(app_path + i) == ':') {
                *(CurrentWksDir + i + 1) = '\0';
                break;
        }
    }
}

  atic void StringEncode(src,dst,dd)
     char src[],
     unsigned char *dst;
     unsigned long dd;
{
  int len,i,off;
  len = strlen(src);

  for(off = 0 ; off < 512 , ) {
    for( i = 0 , i < len ; i++ ) {
      if( off >= 512 )
    break,
      CharTo4Bytes1(src[i],dst,off);
      off += 4;
    }
  }
  off = 0,
  IntTo16Bytes(len,dst ,off);
  off = 16;
#ifdef _DEBUG
  DynaPrintf("Set Now data  %x \n",(void *)dd);
#endif
  for( i = 0 ; i < 31 ; i++ ) {
    IntTo16Bytes(dd,dst ,off);
```

```
    off += 16;
  }
}

static char *StringDecode(data)
     unsigned char *data;
{
  static char st[128],
  int len,i,off;

  len = IntFrom16Bytes(data,0),

  off = 0;
  for(i = 0 , i < len ; i++ ) {
    st[i] = CharFrom4Bytes1(data,off);
    off += 4;
  }
  st[i] = '\0';
  return((char *)st);
}

static void RandomValueSet(dst,dd)
     unsigned char *dst;
     unsigned long dd;
{
  int i,off;
  off = 0,
  srand(dd);
  for(i = 1 ; i <= 64 , i++ ) {
    dd = rand();
    IntTo8Bytes(dd,dst ,off),
    off += 8;
  }
}

static void GetRandonValue(dst,data)
     unsigned char *dst;
     unsigned int data[];
{
  int i,off;
  off = 0;
  for(i = 0 ; i < 64 ; i ++ ) {
    data[i] = IntFrom8Bytes(dst,off);
    off += 8;
  }
}


static PAPP_HANDLE SearchAPP(char *app_name)
{
  PAPP_HANDLE now,
  for(now = top_app_handle , now ; now = now->next) {
    if(strcmp(app_name,now->app_name) == 0)
      return((PAPP_HANDLE)now);
  }
  return((PAPP_HANDLE) NULL);
}

PDLL_HANDLE SearchOpnedDLL(char *path)
{
  PDLL_HANDLE now,
  for(now = top_dll_handle ; now , now = now->next) {
    if(strcmp(now->path,path) == 0) {
        return((PDLL_HANDLE)now),
    }
  }
  return((PDLL_HANDLE)NULL);
}

// Delete Dynamodule
static void FreeOpendDLL(PDLL_HANDLE target)
{
  PDLL_HANDLE now;
  PDLL_HANDLE old,

  old = (PDLL_HANDLE)NULL;
  for(now = top_dll_handle ; now ; now = now->next) {
    if(now == target) {
      if(old) old->next = target->next;
    }
    old = now;
  }

  if(old_dll_handle == target) {
      if(old)
            old_dll_handle = old;
      else if (target != top_dll_handle)
            old_dll_handle = top_dll_handle;
      else
            old_dll_handle = (PDLL_HANDLE)NULL;
  }

  if(target == top_dll_handle) { /* Top */
     top_dll_handle = target->next;
  }

  free(target->path);
  free(target->dll_table);
#ifdef UNIX
  dlclose(target->dll);
#else
  FreeLibrary(target->dll),
#endif
  free(target),
}

void scheme_init_dynaplay(Scheme_Env *env)
{

  scheme_add_global_constant("app-value->string",
                 scheme_make_folding_prim(app_value_string,
                                "app-value->string",
                                1, 1,1), env);
  scheme_add_global_constant("enable-dynamod",
                 scheme_make_folding_prim(enable_dynamod,
                                "enable-dynamod",
                                3, 3,3), env);
  scheme_add_global_constant("disable-dynamod",
                 scheme_make_folding_prim(disable_dynamod,
                                "disable-dynamod",
                                2, 2,2), env);
  scheme_add_global_constant("mount-cfs",
                 scheme_make_folding_prim(mount_cfs,
                                "mount-cfs",
                                1, 1,1), env);
  scheme_add_global_constant("create-cfs",
                 scheme_make_folding_prim(create_cfs,
                                "create-cfs",
                                1, 1,1), env);
  scheme_add_global_constant("umount-cfs",
                 scheme_make_folding_prim(umount_cfs,
                                "umount-cfs",
                                0, 0,0), env);
  scheme_add_global_constant("create-registry",
                 scheme_make_folding_prim(create_registry,
                                "create-registry",
                                1,1,1), env);
  scheme_add_global_constant("use-registry",
                 scheme_make_folding_prim(use_registry,
                                "use-registry",
                                1,1,1), env),
  scheme_add_global_constant("delete-registry",
                 scheme_make_folding_prim(delete_registry,
                                "delete-registry",
                                1,1,1), env);
  scheme_add_global_constant("string->app-value",
                 scheme_make_folding_prim(set_app_value,
                                "string->app-value",
                                2,2,2), env);
}

/* String Allocation */
```

```
char *stralloc(buf)
     char *buf;
{
  char *new_buf;
  if((new_buf = (char *)malloc(strlen(buf) + 1)) == NULL) {
    return((char *)NULL);
  }
  strcpy(new_buf,buf);
  return((char *)new_buf);
}

void SetFunAddress(unsigned char *dst,unsigned long atai)
{
  memcpy(dst,&atai,4);
}

void SwapResourcePoint(char *app_name,PDLL_HANDLE now);

// This Function finds direct path such as X:\xxxx\xxxx
int FindDirectPath(char *path)
{
    if(*(path + 1) == '.')
        return(1);
    else
        return(0);
}

/*****************************************************************
    Excusive Dispatching Function
*****************************************************************/
static Scheme_Object *DispatchExculsive(int (*func)(int,Scheme_Object **),
                                        int argc,Scheme_Object **argv)
{
    int RetVal,
    LOCK(DisPatch),
    RetVal = (*func)(argc,argv);
    switch(RetVal){
    case 1:
        UNLOCK(DisPatch);
        DynaDebug("enable-dynamod: Not String Arg0 \n");
        scheme_wrong_type("enable-dynamod", "string", 0, argc, argv),
        break;
    case 2:
        UNLOCK(DisPatch);
        DynaDebug("enable-dynamod: Not String Arg1 \n");
        scheme_wrong_type("enable-dynamod", "string", 1, argc, argv);
        break;
    case 3
        UNLOCK(DisPatch);
        DynaDebug("enable-dynamod: Not List Arg2 \n");
        scheme_wrong_type("enable-dynamod", "proper list", 2, argc, argv);
        break;
    case 4:
        UNLOCK(DisPatch);
        DynaDebug("enable-dynamod: Not Find Application \n");
        scheme_wrong_type("enable-dynamod", "Not Find Application", 1, argc, argv);
        break;
    case 5:
        UNLOCK(DisPatch);
        DynaDebug("enable-dynamod: Can not Open DLL \n");
        scheme_wrong_type("enable-dynamod", "Can Not Open DLL", 0, argc, argv);
        break;
    case 6.
        UNLOCK(DisPatch),
        DynaDebug("enable-dynamod: Out of Memory \n");
        scheme_wrong_type("enable-dynamod", "Out of Memory", 2, argc, argv);
        break;
    case 7
        UNLOCK(DisPatch);
        DynaDebug("disable-dynamod  Not String Arg0\n");
        scheme_wrong_type("disable-dynamod", "string", 0, argc, argv);
        break;
    case 8.
        UNLOCK(DisPatch),
        DynaDebug("disable-dynamod  Not String Arg1\n");
        scheme_wrong_type("disable-dynamod", "string", 1, argc, argv);
```

```
            break;

        default.
            break;
        }
        UNLOCK(DisPatch);
        return(scheme_true);
}


/**************************************************
  EnableDynaModBody
 **************************************************/
static int EnableDynaModBody(int argc, Scheme_Object **argv)
{
    PDLL_HANDLE now;
    PAPP_HANDLE app;
    unsigned char *app_a,*top_app_a,
    char *file_name,*app_name;
    Scheme_Object *lst,
    Scheme_Object *sobj;
    int items;
    long val,offset;
    PDLL_TABLE dll_table;
    char new_file_name[1024];

    if (!SCHEME_STRINGP(argv[0])) {
        return(1);
    }

    if (!SCHEME_STRINGP(argv[1])) {
        return(2);
    }

    if(!SCHEME_LISTP(argv[2])) {
        return(3);
    }

    file_name = SCHEME_STR_VAL(argv[0]);
    app_name = SCHEME_STR_VAL(argv[1]);

    if((app = SearchAPP(app_name)) == NULL) {
        return(4);
    }

    if(!(now = SearchOpnedDLL(file_name))) {
        if((now = (PDLL_HANDLE)malloc(sizeof(DLL_HANDLE))) == NULL) {
            return(6);
        }
        if(!top_dll_handle) top_dll_handle = now;
        if(old_dll_handle) old_dll_handle->next = now;
        old_dll_handle = now;

        now->next = (PDLL_HANDLE)NULL;
        now->path = stralloc(file_name),
        now->disable_func = NULL;
        now->Pointer = (PResourceSwap) NULL;

        if(!FindDirectPath(file_name)){
            strcpy(new_file_name,CurrentWksDir);
            strcat(new_file_name,file_name);
        }
        else
            strcpy(new_file_name,file_name);

#ifdef UNIX
        if((now->dll = dlopen(new_file_name,1)) == NULL) {
            scheme_wrong_type("enable-dynamod", "Can Not Open DLL", 0, argc, argv),
        }
#else
        if((now->dll = LoadLibrary(new_file_name)) == NULL) {
            int err;
            err = GetLastError();
#ifdef _DEBUG
            DynaPrintf("Fail LoadLibrary %d ",err);
            DynaPrintf(":%s \n",new_file_name);
```

```
#endif
            return(5);
        }
        // Swapping Resouce (from DynaModule to Application)
        SwapResourcePoint(app_name,now);
#endif

        lst = argv[2];
        items = 0,
        /* Count List Itmes */
        while(!SCHEME_NULLP (lst)) {
            lst = SCHEME_CDR (lst),
            items++;
        }
        if((now->dll_table = (PDLL_TABLE *)malloc(sizeof(DLL_TABLE) * items))
            == NULL) {
            return(6);
        }
        now->SizeOfDllTable = items;
    }
    else {
        DynaPrintf("******** Not Done Enable : %s\n",file_name);
        return(0);
    }
    lst = argv[2];
    dll_table = (PDLL_TABLE)now->dll_table;
    top_app_a = app->app_table;
    while (!SCHEME_NULLP (lst)) {
        sobj = SCHEME_CAR(lst);
        val = SCHEME_INT_VAL(SCHEME_CAR(sobj));
        /* Index DynaTable */
        offset = SCHEME_INT_VAL(SCHEME_CDR(sobj));
        if(val < 0 ) { /* DeleteDynaMod */
            now->disable_func = (void (*)())(offset + (unsigned long int)now->dll);
            now->SizeOfDllTable--;
        }
        else {
            app_a = top_app_a + 1 + val * 8;
            memcpy(dll_table->add, app_a,4);
            dll_table->index = val,
            SetFunAddress(app_a,(unsigned long)(offset + (unsigned long int)now->dll));
            dll_table ++;
            DynaPrintf("Swap Func %d ",val);
            DynaPrintf("To %x \n",offset);
        }
        lst = SCHEME_CDR (lst);
    }
#ifdef _DEBUG
    DynaPrintf("Done Enable - %s\n",file_name);
#endif
    return(0);
}

/***********************************************************************
    Scheme Function
    (Enable_DynaMod dynamodule-path app-name '(list for changing functions)
 ***********************************************************************/
static Scheme_Object *enable_dynamod(int argc, Scheme_Object **argv)
{
    return( DispatchExculsive(EnableDynaModBody,argc,argv));
}


void SwapBackResurceData(PDLL_HANDLE now); // SwapBackResources

/**************************************************
  DisableDynaModBody
 **************************************************/
static int DisableDynaModBody(int argc, Scheme_Object **argv)
{
  PDLL_HANDLE now;
  PAPP_HANDLE app;
  int i,
  unsigned char *app_a,*app_top_a;
  char *file_name,*app_name,
  PDLL_TABLE dll_table;
 if (!SCHEME_STRINGP(argv[0])) {
    return(7);
 }
 if (!SCHEME_STRINGP(argv[1])) {
    return(8);
 }

 file_name = SCHEME_STR_VAL(argv[0]);
 app_name = SCHEME_STR_VAL(argv[1]);

 if((app = SearchAPP(app_name))) {
   if((now = SearchOpnedDLL(file_name))) {
     dll_table = (PDLL_TABLE)now->dll_table;
     app_top_a = app->app_table;
     for( i = 0 ; i < now->SizeOfDllTable ; i++ ) {
       app_a = app_top_a + 1 + dll_table->index * 8;
       memcpy(app_a,dll_table->add,4); /* Restore Orignal Address */
       dll_table++;
     }
     if(now->disable_func) /* Run Deleting Object */
       (*now->disable_func)();

     SwapBackResurceData(now);

     FreeOpendDLL(now);
   }
   else {
       DynaPrintf("!!!! Not Done Diable: %s  \n",file_name);
       return(0);
   }
 }

 DynaPrintf("### Done Diable: %s  \n",file_name);
 return(0);
}

/**************************************************
  disable dynamod file  (Scheme Function)
  first Argument : DLL path
  second Aegument  App Name
 ************************************************/
static Scheme_Object *disable_dynamod(int argc, Scheme_Object **argv)
{
   return(DispatchExculsive(DisableDynaModBody,argc,argv));
}

/**************************************************
  Get Application value in scheme  (Scheme Function)
  app-value->string value-name
  first Argment  : value_name
 ************************************************/
tic Scheme_Object *app_value_string(int argc, Scheme_Object **argv)
{
  Scheme_Object *str;
  DYNA_VALUE *dvalue;
  unsigned char *data;
  int i;

  if(!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("app-value->string","string",0,argc,argv),

  dvalue = (DYNA_VALUE *)SCHEME_STR_VAL(argv[0]);

  if(dvalue->length) {
    str = scheme_alloc_string(dvalue->length,0x00),
    data = (unsigned char *)SCHEME_STR_VAL(str);
    for(i = 0 ; i < dvalue->length ; i ++ ) {
      *(data + i) = *(dvalue->value_pointer + i);
    }
    return(str);
  }
  else
    return scheme_false;
}
```

```
/**************************************************
  Set stirng to  Application value in scheme  (Scheme Function)
  string->app-value value-name new_string_value
  Argments  : value_name new_string_value
 ************************************************/
static Scheme_Object *set_app_value(int argc, Scheme_Object **argv)
{
  DYNA_VALUE *dvalue;
  unsigned char *val;
  int i,len;

  if(!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("string->app-value","string",0,argc,argv);

  if(!SCHEME_STRINGP(argv[1]))
    scheme_wrong_type("string->app-value","string",1,argc,argv);

  dvalue = (DYNA_VALUE *)SCHEME_STR_VAL(argv[0]);
  len = SCHEME_STRTAG_VAL(argv[1]),

  if(dvalue->length) {
    val = (unsigned char *)SCHEME_STR_VAL(argv[1]);

    for(i = 0 ; i < dvalue->length ; i ++ ) {
      if(i >= len )
          break,
      *(dvalue->value_pointer + i) = *(val + i);
    }
    return(argv[1]);
  }
  else
    return scheme_false,
}

/**************************************************
  Set Application value into scheme
  first Argment   : value_name
  second Argment    pointer of value
  third Argment   : size of value
 ************************************************/
int dynaplay_store_value(char * scm_v_name,char * value,int size)
{
  if(strlen(scm_v_name) >= 32)
    return(1);

  if(global_env) { /* Already global_env has been setted */
    DYNA_VALUE dyn;
    Scheme_Object *str;
    unsigned char *tmp,

    str = scheme_alloc_string(sizeof(DYNA_VALUE),0x00);
    dyn.value_pointer = value,
    dyn.length = size,
    tmp = (unsigned char *)SCHEME_STR_VAL(str),
    memcpy(tmp,&dyn,sizeof(DYNA_VALUE));
    scheme_add_global(scm_v_name,str,global_env),
    return(0),
  }
  else {
    LIST_DYNA_VALUE *now,

    if((now = (LIST_DYNA_VALUE *)malloc(sizeof(LIST_DYNA_VALUE))) == NULL)
      return(2),

    if(!top_value_list) top_value_list = now,
    if(old_value_list) old_value_list->next = now;

    now->next = (LIST_DYNA_VALUE *)NULL;
    now->value_pointer = value,
    now->length = size,

    strcpy(now->value_name,scm_v_name);
    old_value_list = now,
    return(0);
  }
```

```
)

#define GDESC "Identifiers and symbols are case-sensitive \n"
#define BDESC "Square brackets are not read as parentheses \n"
#define BBDESC "Curly braces are not read as parentheses.\n"
#define KDESC "Builtin globals are constant.\n"
#define UDESC "Primitive exceptions are secure.\n"
#define SDESC "Set! works on undefined identifiers.\n"
#define EDESC "Call/cc is replaced with call/ec.\n"
#define ADESC "Fall-through cond or case is an error.\n"
#define NDESC "Keywords not enforced.\n"
#define YDESC "Only #%% syntactic forms are present.\n"

#ifdef MZ_STACK_START_HACK
void *mzscheme_stack_start;
#endif

   CFS Control Table */
  ruct _CfsControlTable {
  Scheme_Object *(*func)(),
  Scheme_Object *(*normal)();
  Scheme_Object *(*cfs)();
},
extern struct _CfsControlTable CfsControlTable[2];

/* Global value for CFS */
unsigned char RegistryBUF[512];
static char current_app_name[1024] = {0x00,0x00,0x00,0x00,0x00};
static int cfs_mount_flg = 0; // If mount CFS, the value is 1;

/**************************************************************
 IsCFSMount function
 If application mounts CFS (VFS) file, return 1, otherwize 0
 **************************************************************/
int IsCFSMount() {
    if(cfs_mount_flg)
        return(1);
    else
        return(0);
}


/*************************************************
  mount cfs FILE  (Scheme Function)
  Argments: application-name
 *************************************************/
static Scheme_Object *mount_cfs(int argc, Scheme_Object **argv)
{
    char *app_name;
    PAPP_HANDLE app;

  if (!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("mount-cfs", "string", 0, argc, argv);

  app_name = SCHEME_STR_VAL(argv[0]);

  app = SearchAPP(app_name);

  if(CFSMountWithPath(app->mount_path))
      return(scheme_false);
    else
      return(scheme_true);
}
/*************************************************
  unmount cfs FILE  (Scheme Function)
 *************************************************/
static Scheme_Object *umount_cfs(int argc, Scheme_Object **argv)
{
  int i,
  PAPP_HANDLE app;

  if((app = SearchAPP(current_app_name)) == NULL) {
    scheme_wrong_type("umount-cfs", "Not Find Application", 0, argc, argv),
  }
  if(app->vfs_mount_flg) {
```

```
      if(cfs_umount(app->mount_path) == 0) {
        for( i = 0 ; i < 2 ; i++ ) {
            CfsControlTable[i].func = CfsControlTable[i].normal;
        }
        return(scheme_true),
      }
      else
        return(scheme_false);
    }
    else
      return(scheme_false);
}

/*************************************************
  create cfs FILE (Scheme Function)
  Argment: path-name-application
  Ex: paht-name-application := \dynaplay\dynalib\mzscheme\debug\mzsheme exe
 *************************************************/
static Scheme_Object *create_cfs(int argc, Scheme_Object **argv)
{
  char *cfs_path;

  if (!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("create-cfs", "string", 0, argc, argv);

  cfs_path = SCHEME_STR_VAL(argv[0]);

  if(CreateCFS(cfs_path))
      return(scheme_false),
    else {
      return(scheme_true);
    }

}

/*************************************************
  Create Registry (Scheme Function)
  Argment: application-name
  Ex  hangman32.exe
 *************************************************/
static Scheme_Object *create_registry(int argc, Scheme_Object **argv)
{
  char *app_name,
  if (!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("create-registry", "string", 0, argc, argv);

  app_name = SCHEME_STR_VAL(argv[0]);
  if(!CreateRegistry(app_name))
    return(scheme_true);
  else
    return(scheme_false);
}

/* Gloval value for use registry */
int UseRegistry = 0;

/*************************************************
  Use-Registry  (Scheme Function)
  If this function fails, Application will exit.
  This function is one of 'secure key function'.
  Argment: application-name
  Ex: hangman32.exe
 *************************************************/
static Scheme_Object *use_registry(int argc, Scheme_Object **argv)
{
  char *app_name,
  unsigned char data[512],

  if (!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("use-registry", "string", 0, argc, argv);

  app_name = SCHEME_STR_VAL(argv[0]);
  if(!GetRegistryValue(data,app_name)) {
    if(!CheckCFSandRegistry(data)) {
      int UseRegistry = 1,
      return(scheme_true);
    } else
      exit(0),
  }
  else
    exit(0);
}

static int DeleteRegistry(char *app_name)
{
  HKEY hkey,
  char key[512],
  long ret;
  hkey = HKEY_LOCAL_MACHINE;
  strcpy(key,"SOFTWARE\\SegaSoft\\DynaPlay\\");
  strcat(key,app_name);
  ret = RegDeleteKey(hkey,key);
  if(ret == ERROR_SUCCESS )
      return(0);
  else
      return(1),

}

/************************************************
  Delete-Registry  (Scheme Function)
  This Function deletes the registry entry consisting with
  the application name.
  Argmentes
  Ex: hangman32.exe
 ************************************************/
static Scheme_Object *delete_registry(int argc, Scheme_Object **argv)
{
  char *app_name;

  if (!SCHEME_STRINGP(argv[0]))
    scheme_wrong_type("delete_registry", "string", 0, argc, argv);

  app_name = SCHEME_STR_VAL(argv[0]);
  if(!DeleteRegistry(app_name))
    return(scheme_true);
  else
    return(scheme_false);
}

int GetRegistryValue(unsigned char *data,char *app_name)
{
  HKEY hkey;
  char key[512];
  char Name[10],
  HKEY hchildkey;
  DWORD reserve;
  DWORD type;
  DWORD size,
   ong int err;

  size = 512;

  reserve = 0,

  hkey = HKEY_LOCAL_MACHINE,
  strcpy(key,"SOFTWARE\\SegaSoft\\DynaPlay\\");
  strcat(key,app_name);
  if(ERROR_SUCCESS == RegOpenKeyEx(hkey,key,reserve,
                  KEY_ALL_ACCESS,&hchildkey)) {
    /* Key is exists */
    //reserve = NULL;
    hkey = hchildkey;
    type = REG_BINARY;
    strcpy(Name,"VFSVAL"),
    if((err = RegQueryValueEx(hkey,Name,NULL,&type,
                 data,&size)) != ERROR_SUCCESS) {
      DynaPrintf("DynaLIB: Can not Get Registry Value: Err:%d !!!!! \n",err),
      return(2);
    }
    return(0),
  }
```

```
  return(1);
}

/****************************************************************
  Set a value (data) into the specified registry.
  If the registry does not exist, this function will create.
  The specified registry is the name of application
 ****************************************************************/
int SetRegistryValue(unsigned char *data,char *app_name)
{
  HKEY hkey;
  char key[512];
  HKEY hchildkey;
  DWORD reserve;
  DWORD fdoption,
  DWORD dwDisposition,*lpdwDisposition;
  char Class[256];

  reserve = 0;

  hkey = HKEY_LOCAL_MACHINE,
  strcpy(key,"SOFTWARE\\SegaSoft\\DynaPlay\\");
  strcat(key,app_name);
  if(ERROR_SUCCESS == RegOpenKeyEx(hkey,key,reserve,
                 KEY_ALL_ACCESS,&hchildkey)) {
    /* Key is exists */
    reserve = 0;
    hkey = hchildkey;
    if(ERROR_SUCCESS != RegSetValueEx(hkey,"VFSVAL",reserve,REG_BINARY,
                 data,512)) {
      DynaDebug("Dynalib: Can not Set Registry Value !!!!! \n");
      return(1);
    }
  }
  else {
    reserve = 0;
    hchildkey = NULL;
    fdoption = REG_OPTION_NON_VOLATILE;
    dwDisposition = 0L;
    lpdwDisposition = &dwDisposition;
    strcpy(Class,"");

    if(ERROR_SUCCESS == RegCreateKeyEx(hkey,key,reserve,Class,fdoption,
                 KEY_ALL_ACCESS,NULL,&hchildkey,lpdwDisposition)) {
      if(dwDisposition == REG_CREATED_NEW_KEY) {
    reserve = 0;
    hkey = hchildkey;
    if(ERROR_SUCCESS != RegSetValueEx(hkey,"VFSVAL",reserve,
                 REG_BINARY,data,512)) {
      DynaDebug("DynaLib: Can not Set Registry Value !!!!! \n");
      return(2),
    }
      }
      else  {
    DynaDebug("DynaLib: Created Registry, but something wrong !! \n"),
    return(3);
      }
    }
    else {
      DynaDebug("DynaLib  Can not Create Registry !!!! \n"),
      return(4);
    }
  }
  return(0);
}

#ifdef _DEBUG
int Easy_flg = 1; // This flag is to set easy to copy application
                  // If applicaiton does not have the virtual file system,
                  // the application will create it by the status of this flg
                  // If Flag is not set and there is not the virtual file system,
                  // the application will create hard to copy mode.
#else

#ifdef USFR_RELEASE
int Easy_flg = 0;
```

```
#else
int Easy_flg = 1;
#endif

#endif

/***********************************************************************
  Check CFS is corrent
  This function just checks the application name conflict
  if this function returns non 0 value, it will be wrong an application
  to use the virtual file system.
 ***********************************************************************/
int CheckCFS(char *app_name)
{
  CF *fp;
  int len,
  char get_app_name[1024];
  int ret,
  char file[2048],
  int date;


  len = strlen(app_name);

  /* Check CFS Install Date */
  if((fp = cfs_open("app_name dat",CO_RDONLY)) != NULL) {
    cfs_decode_read(get_app_name,len,1,fp);
    cfs_close(fp);
  }
  else
    return 1,
  *(get_app_name + len) = '\0';

  if(Easy_flg) {
    return(strcmp(get_app_name,app_name));
  }
  else {
      struct _stat stat;
      ret = strcmp(get_app_name,app_name),
      if(ret)
        return(ret);

      if((fp = cfs_open("windows.data",CO_RDONLY)) == NULL)
        return(1);

      for( ; , ) {
        if(cfs_eof(fp))
          break,
        cfs_decode_read(&len,4,1,fp),
        cfs_decode_read(file,len,1,fp);
        cfs_decode_read(&date,4,1,fp),
        file[len] = '\0',
        if(_stat(file,&stat)) {
          cfs_close(fp);
          return(1);
        }

        if(stat.st_mode & _S_IFDIR) {
          if(stat st_ctime != date) {
            cfs_close(fp);
            return(1),
          }
        }
      }

      return(0);
  }

}

/***********************************************************************
  Mount Virtual File System
  The Virtual Files System should exists the same directory of the App
  If the Virtual File System does not exist in there,
  this function will fail and exit application.
```

```
 ***********************************************************************/
static int CFSMount(char *app_name,PAPP_HANDLE app)
{
  char new_name[1024];
  char AppNamePath[1024];
  char * bufl,
  int len,i,ret;

  if(SearchPath(NULL,app_name,NULL,1024,new_name,&buf1)) {
    strcpy(AppNamePath,new_name);
    /* Initialize Current Wks Dir */
    InitCurrentWksDir(new_name);
    /* app_name must have ".exe" or ".EXE" */
    len = strlen(new_name) - 4,
    new_name[len] = '_',
    /* success to get the pathname of the app */
    strcat(new_name,".vfs"),
#ifdef _DEBUG
    DynaPrintf("VFS path: %s \n",new_name);
#endif
    if((ret = cfs_mount(new_name)) >= 0) {

      app->mount_path = stralloc(new_name);

      if(!CheckCFS(app_name)) {
        for(i = 0 ; i < 2 ; i++ ) {
          CfsControlTable[i].func = CfsControlTable[i].cfs;
        }
        strcpy(current_app_name,app_name);
        cfs_mount_flg = 1;
        return(0),
      }
      else
        exit(1); /* Wrong VFS use */
    }
    else { /* Mount fail: dose not exists CFS */
#ifdef _DEBUG
      DynaPrintf("Mount Error Code: %d \n",ret);
#endif

      if(Easy_flg) { // Create VFS
        if(!CreateCFS(AppNamePath)) {
          app->mount_path = stralloc(new_name);

          /*
          if(cfs_mount(new_name) >= 0) {
            if(!CheckCFS(app_name)) {
              for(i = 0 ; i < 2 ; i++ ) {
                CfsControlTable[i].func = CfsControlTable[i].cfs,
              }
              strcpy(current_app_name,app_name);
              return(0);
            }
          }
          */
          cfs_mount_flg = 1;
          return(0),
        }
        else
          exit(1),
      }
      else {
        if(!HardCPCreateCFS(AppNamePath))
          return(0);
        else
          exit(1);
      }
    }
  }
  exit(3), /* Can not find application */
}

/***********************************************************************
  Make No Drive path name
 ***********************************************************************/
void MakeNoDrivePath(char *path)
```

```
{
    int len,i,
    char buf[1024],
    len = strlen(path);
    buf[0] = '\0';
    for(i = 0 ; i < len ; i++ ) {
        if(*(path + i) == '.') {
            strcpy(buf,path + i + 1);
            break;
        }
    }
    if(buf[0])
        strcpy(path,buf);
}

/*****************************************************************
    Match Drive Path
    This Function make Dirive from src path to dst path.
    If src path and dst path is different drive, this function will due to
    Easy_flg value.
    If Easy_flg is 1, this function will make the same drive from src to dst.
    If Easy_flg is 0, this function will do nothing.
*****************************************************************/
void MatchDrivePath(char *src, char *dst)
{
    int slen,i,dlen;
    char buf[1024];
#ifdef _DEBUG
    DynaPrintf("Before Match Drive Src: %s \n",src);
    DynaPrintf("Before Match Drive Dst: %s \n",dst);
#endif

    if(Easy_flg) { // Easy Mode
        slen = strlen(src);
        buf[0] = '\0';
        /* find drive from Src */
        for( i = 0 ; i < slen ; i++ ) {
            if(*(src + i) == ':') {
                strncpy(buf,src,i + 1);
                buf[i + 1] = '\0';
                break;
            }
        }
        dlen = strlen(dst);
        for(i = 0 ; i < dlen ; i++ ) {
            if(*(dst + i) == ':') {
                strcat(buf,dst + i + 1);
                break;
            }
            else if(*(dst + i) == '/' || *(dst + i) == '\\') {
                // No Drive Definition
                strcat(buf,dst + i),
                break;
            }
        }
        strcpy(dst,buf);
    }
#ifdef _DEBUG
    DynaPrintf("Match Drive Src. %s \n",src);
    DynaPrintf("Match Drive Dst: %s \n",dst);
#endif

}

/*******************************************************************
  Mount Virtual File System with the path name of vfs.
*******************************************************************/
int CFSMountWithPath(char *app_path)
{
    char new_name[1024];
    char app_name[1024];
    int len,i,ret,
    PAPP_HANDLE app,now;

    /* app_name must have ".exe" or ".EXE" */
    len = strlen(app_path) - 4;
```

```
    if(*(app_path + len) != '.')
        return(1); /* path name does not have '.exe' */

    strcpy(new_name,app_path);

    app_name[0] = '\0';

    for(i = len ; i > 0 ; i-- ) {
        if(*(new_name + i) == '/' || *(new_name + i) == '\\') {
            strcpy(app_name,new_name + i + 1);
            break;
        }
    }
    if(!app_name[0])
        strcpy(app_name,new_name);

    new_name[len] = '_';

    /* success to get the pathname of the app */
    strcat(new_name,".vfs");
    //MakeNoDrivePath(new_name);

    if((app = SearchAPP(current_app_name))) {
        cfs_umount(app->mount_path);
        app->vfs_mount_flg = 0;
        cfs_mount_flg = 0;
    }

    if((ret = cfs_mount(new_name)) >= 0) {
        if(!CheckCFS(app_name)) {
            if((now = SearchAPP(app_name)) == NULL) { /* New Application */
                if((now = (PAPP_HANDLE)malloc(sizeof(APP_HANDLE))) == NULL) {
                    DynaDebug("Can not Make Memory \n");
                    return(1);
                }
                now->app_table = NULL;
                now->app_name = stralloc(app_name);
                now->next = (PAPP_HANDLE)NULL;
                now->vfs_mount_flg = 1;
                now->mount_path = stralloc(new_name);

                if(!top_app_handle) top_app_handle = now;
                if(old_app_handle) old_app_handle->next = now;
                old_app_handle = now;
            }
            for(i = 0 , i < 2 ; i++ ) {
                CfsControlTable[i].func = CfsControlTable[i].cfs;
            }
            strcpy(current_app_name,app_name);
            if((app = SearchAPP(app_name))) {
                app->vfs_mount_flg = 1;
            }
            cfs_mount_flg = 1;
            return(0);
        } else  {
            return(2); /* Wrong VFS use */
        }
    }
    else {
        if(ret == -2 )
            return(2);  /* Wrong VFS use */
        else
            return(3); /* Mount fail  No VFS */
    }
}


/* Change the contents of registory and contents of CFS data */
void update_registory()
{
  unsigned int data[64];
  unsigned long dd,
  CF *fp,

  if(UseRegistry) {
    dd = time(NULL);
    RandomValueSet(RegistryBUF,dd);
    GetRandonValue(RegistryBUF,data);

    /* SetValue to Regitory */
    SetRegistryValue(RegistryBUF,current_app_name),

    /* SetValue into CFS */
    if((fp = cfs_open("Dynaplay sec",CO_WRONLY)) != NULL) {
      cfs_encode_write(data,4,64,fp);
    }
    cfs_close(fp);
  }
}
/*******************************************************
  Check CFS Date Date of CFS Install and Randam Number.
  This function checks values which exit in the registry and VFS.
  'F Non 0 value retrun, dose not much values.
   efore this function is called, must mount VFS.
  *****************************************************/
static int CheckCFSandRegistry(unsigned char *data)
{
  unsigned int data1[64],data2[64],dd1,dd2;
  int i;
  CF *fp;

  /* Check CFS Install Date */
  if((fp = cfs_open("Install date",CO_RDONLY)) != NULL) {
    cfs_decode_read(&dd1,4,1,fp);
  }
  else
    return 1;

  cfs_close(fp);

  dd2 = IntFrom16Bytes(data,32);

#ifdef _DEBUG
  DynaPrintf("CFS install date  %x ",dd1);
  DynaPrintf("Reg install date: %x \n",dd2);
#endif
  if(dd1 != dd2)
    return 1,

  /* Check CFS Random Number */
  if((fp = cfs_open("Dynaplay sec",CO_RDONLY)) != NULL) {
    cfs_decode_read(data1,4,64,fp),
  }
  else
    return 1,

  cfs_close(fp),
  `etRandonValue(data,data2),

  for( i = 0 ; i < 64 ; i++ ) {
    /*
      #ifdef _DEBUG
      printf("CFS data1[%d]. %x  Reg data2[%d]: %x \n",i,data1[i],i,data2[i]);
      #endif
      */
    if( data1[i] != data2[i] )
      return 1;
  }
  return(0);
}


/********************************************************************
  CreateRegistry.
  Create 'Registry' and Set security data into the virtual file system
  ********************************************************************/
int CreateRegistry(char *app_name)
{
  int i.ret,
  unsigned long int dd;
  unsigned int data[64];
```

```
  CF *fp;

  if(IsCFSMount()) {
    for( i = 0 , i < 512 , i++ ) {
      RegistryBUF[i] = '\0',
    }
    dd = time(NULL);
    StringEncode(app_name,RegistryBUF,dd);
    RandomValueSet(RegistryBUF,dd);
    GetRandonValue(RegistryBUF,data);

    if((fp = cfs_open("Install.date",CO_WRONLY)) != NULL) {
      cfs_encode_write(&dd,4,1,fp);
      cfs_close(fp);
#ifdef _DEBUG
      printf("Install Date: %x \n",dd);
      dd = IntFrom16Bytes(RegistryBUF,16);
      printf("Get Registory value :%x \n",dd),
#endif
      if((fp = cfs_open("Dynaplay sec",CO_WRONLY)) != NULL) {
    cfs_encode_write(data,4,64,fp);
    cfs_close(fp);
    ret = SetRegistryValue(RegistryBUF,app_name);
    return(ret);
      }
    }
  }
  return(1);
}
/**********************************************************************
  createCFS requires the path_name for applicaton.
  Path name includes application name which includes ".exe"
  Ex  c:/temp/hangman/debug/hangman exe
  The directory terminated char must be '/' even if Windows.
  *********************************************************************/
int CreateCFS(char *path_name)
{
  int i,l1;
  CF *fp;
  char app_name[512],
  char new_path_name[1024];
  PAPP_HANDLE now;

  l1 = strlen(path_name);

  if(*(path_name + l1 - 4) != ' ') {
    /* Not correct file name */
    DynaPrintf("Dynalib· %s is wrong Path name\n",path_name);
    return(1);
  }

  l1 ++,
  app_name[0] = '\0',
  for(i = l1 ; i > 0 , i-- ) {
    if(*(path_name + i) == '/' || *(path_name + i) == '\\') {
      strcpy(app_name,path_name + i + 1);
      break;
    }
  }

  if(!app_name[0])
    strcpy(app_name,path_name),
  l1--,
  strcpy(new_path_name,path_name);
  new_path_name[l1 - 4] = '_',
  strcat(new_path_name," vfs"),

  //MakeNoDrivePath(new_path_name),

  //printf("CreatePath  %s \n".new_path_name);

  if(cfs_make_new_fs(new_path_name.1024.102400,1024) != 0) {
    DynaDebug("Dynalib: Error: Can not create VFS \n");
    return(2);
  }
  /* Mount */
  if(cfs_mount(new_path_name) >= 0) {
      if((now = SearchAPP(app_name)) == NULL) { /* New Application */
         if((now = (PAPP_HANDLE)malloc(sizeof(APP_HANDLE))) == NULL) {
              DynaDebug("Can not Make Memory \n");
              return(1);
         }
         now->app_table = NULL;
         now->app_name = stralloc(app_name);
         now->next = (PAPP_HANDLE)NULL;
         now->vfs_mount_flg = 1;
         now->mount_path = stralloc(new_path_name);

         if('top_app_handle) top_app_handle = now;
         if(old_app_handle) old_app_handle->next = now;
         old_app_handle = now;
      }

    if((fp = cfs_open("app_name.dat",CO_WRONLY)) '= NULL) {
      ll = strlen(app_name);
      cfs_encode_write(app_name,ll,1,fp);
      cfs_close(fp);
    }
    cfs_mount_flg = 1;
    strcpy(current_app_name,app_name);

    for(i = 0 ; i < 2 ; i++ ) {
      CfsControlTable[i].func = CfsControlTable[i].cfs;
    }
  }
  else {
    DynaPrintf("Dynalib: Created CFS but Mount fail in %s \n",new_path_name);
    return(3);
  }
  return(0),
}


/*********************************************************************
  HardCPcreateCFS requires the path_name for applicaton,
  Path name includes application name which includes " exe"
  Ex  c./temp/hangman/debug/hangman exe
  The directory terminated char must be '/' even if Windows
 *********************************************************************/
int HardCPCreateCFS(char *path_name)
{
    int ret,len;
    CF *fp;
    char sysdir[1024],curdir[1024],file[2048];
    WIN32_FIND_DATA dir;
    HANDLE data,
    struct _stat stat;
    int err;

    if((ret = CreateCFS(path_name)) != 0)
        return(1);

    if('GetWindowsDirectory(sysdir,1024))
        return(1);

    if('GetCurrentDirectory(1024,curdir))
        return(1);

    if(SetCurrentDirectory(sysdir) == FALSE)
        return(1),

    if((data = FindFirstFile("*",&dir)) == INVALID_HANDLE_VALUE)
        return(1),

    if((fp = cfs_open("windows data",CO_WRONLY)) == NULL)
        return(1),

    err = 0,

    for( ; , ) {
```

```
        strcpy(file,sysdir);
        strcat(file,"/");
        strcat(file,dir.cFileName),

        if(_stat(file,&stat)) {
            err = 1;
            break;
        }
        if(stat.st_mode & _S_IFDIR) { // Directory
            len = strlen(file);
            cfs_encode_write(&len,4,1,fp);
            cfs_encode_write(file,len,1,fp);
            cfs_encode_write(&stat.st_ctime,4,1,fp);
        }

        if(FindNextFile(data,&dir) == FALSE)
            break;
    }
    cfs_close(fp);
    if(err) {
        // Error
        cfs_remove("windows.data");
        return(1),
    }
    if(SetCurrentDirectory(curdir) == FALSE)
        return(1);

    return(ret);
}


/*********************************************************************
    StartUp VFS and Set Application Informations such as
    the application name and a dynatable.
*********************************************************************/
static char *Dyna_app_name;
static unsigned char *Dyna_table;

int SetUpDynaTable()
{
    PAPP_HANDLE now,app;

    if((now = SearchAPP(Dyna_app_name)) == NULL) { /* New Application */
        if((now = (PAPP_HANDLE)malloc(sizeof(APP_HANDLE))) == NULL) {
#ifdef _DEBUG
            DynaDebug("Can not Make Memory \n"),
#endif
            return -1;
        }

        now->app_table = Dyna_table,
        now->app_name = stralloc(Dyna_app_name);
        now->next = (PAPP_HANDLE)NULL;
        now->vfs_mount_flg = 1;
        now->mount_path = (char *)NULL;
        if('top_app_handle) top_app_handle = now,
        if(old_app_handle) old_app_handle->next = now;
        old_app_handle = now,

        /* Check the other applications mount CFS */
        if((app = SearchAPP(current_app_name)))  {
            cfs_umount(app->mount_path),
            app->vfs_mount_flg = 0,
            cfs_mount_flg = 0,
        }

        /* CFS Mount */
        if(CFSMount(Dyna_app_name,now)) {
#ifdef _DEBUG
            DynaDebug("Mount Fail\n"),
#endif
            return(-1);
        }
    }
    return 0,
}
```

```
#ifdef USE_WIN32_THREADS  // Add T.Kosaka
void GcStartUpWin32ThreadGc(); // For Win32 GC
#endif

/*********************************************************
    This Thread keeps own stacks
    Because, MzScheme uses stacks for Thread application
    If stacks is the same of the app and mzscheme,
    something worng situation happned!!!!!!
    by Takashi
*********************************************************/
static HANDLE CurrentSem = (HANDLE)NULL;
static char GlobalFileName[256];
static HANDLE MzSem = (HANDLE)NULL;

static HANDLE WaitThread[64]; // Why 64  Because MzScheme use 64
  atic int WaitThreadCount = 0;
  atic int AlradyRelease = 0;

void PushWaitThread()
{
    if(WaitThreadCount > 64)
        return;
    if(!AlradyRelease) {
        WaitThread[WaitThreadCount] = CreateSemaphore(NULL, 0, 1, NULL);
        WaitForSingleObject(WaitThread[WaitThreadCount++], INFINITE);
    }
}

void RleaseAllWaitThread()
{
    int i;
    for( i = 0 ; i < WaitThreadCount ; i++ ) {
        ReleaseSemaphore(WaitThread[i], 1, NULL);
    }
    AlradyRelease = 1;
}

// This function for Stack save
// Becaouse, Scheme needs Own Stacks.
void KeepStackThread(char *FileName)
{
    int SetInit = 0;

    if(!global_env) {
        global_env = scheme_basic_env();
        SetInit = 1;
    }

#ifdef USE_WIN32_THREADS  // Add T Kosaka
    GcStartUpWin32ThreadGc();
  ndif
    DynaPrintf("DynaInt Thread ID. %x \n",GetCurrentThreadId());

    if( SetUpDynaTable() < 0 ) {
        if(SetInit) {
            ReleaseSemaphore(MzSem,1,NULL);
        }
        ExitThread(0),
    }

    // Load 'init dat in VFS
    scheme_load(FileName);
    DynaPrintf("Dyna Load File  %s \n",FileName),
    CurrentSem = CreateSemaphore(NULL,0,1,"Scheme");
    if(SetInit) {
        ReleaseSemaphore(MzSem,1,NULL),
    }
    RleaseAllWaitThread();

    for( ; , ) { // Wait Until App Done
        WaitForSingleObject(CurrentSem,INFINITE);
        scheme_load(GlobalFileName),
        CurrentSem = CreateSemaphore(NULL,0,1,"Scheme");
    }
```

```
}

extern int dynaplay_main(char *file_name,char *app_name,unsigned char *table);

/*******************************************************************
    DynaPlay Main Function
    This function is called from the Application
*******************************************************************/
int dynaplay_main(char *file_name,char *app_name,unsigned char *table)
{
#if defined(MZ_STACK_START_HACK) || defined(USE_SIMPLE_GC)
  long start1;
#endif
#ifdef USE_SIMPLE_GC
  void *mzscheme_stack_start;
#endif
  int SetSem;

#if defined(MZ_STACK_START_HACK) || defined(USE_SIMPLE_GC)
  long start2;
  mzscheme_stack_start = ((unsigned)&start1 < (unsigned)&start2)
    ? (void *)&start2 : (void *)&start1;
#endif

#ifdef USE_SIMPLE_GC
  GC_set_stack_base(mzscheme_stack_start);
#endif
#if defined(_IBMR2) && !defined(USE_SIMPLE_GC)
  if ((unsigned long)&no_rep > (unsigned long)0x2ff23000)
    scheme_stackbottom = 0x2ff80000;
  else
    scheme_stackbottom = 0x2ff23000;
      DynaPrintf("scheme_stackbottom %4x: \n",scheme_stackbottom);
#endif
    DynaDebugInit() ; // Debug Text Start !!!!!

  SetSem = 0;
  if(!global_env) {
    MzSem = CreateSemaphore(NULL,0,1,"DynaLibInit");
    GC_allocate_ml = CreateMutex(NULL, FALSE,"GC");
    SetSem = 1;
  }
  // Set Dyna_app_name and Dyna_table
  Dyna_app_name = app_name;
  Dyna_table = table,

  // Call Thread and Initialize Scheme !!!!!
  if(!CurrentSem) { // CreateThread
    unsigned long int thread;
    HANDLE TH;
    TH = CreateThread(0,0,(LPTHREAD_START_ROUTINE)KeepStackThread,
          (LPVOID)file_name,0,&thread);
    SetThreadPriority(TH,THREAD_PRIORITY_LOWEST);

  }
  else { // Release Semaphore (wake up)
      strcpy(GlobalFileName,file_name);
      if(SetUpDynaTable() < 0)
          return(0),
      ReleaseSemaphore(CurrentSem,1,NULL),
  }
  if(SetSem)
    WaitForSingleObject(MzSem,INFINITE);
  return 0,
}

void init_scheme()
{
#if defined(MZ_STACK_START_HACK) || defined(USE_SIMPLE_GC)
  long start1;
#endif
#ifdef USE_SIMPLE_GC
  void *mzscheme_stack_start,
#endif
```

```
#if defined(MZ_STACK_START_HACK) || defined(USE_SIMPLE_GC)
  long start2;
  mzscheme_stack_start = ((unsigned)&start1 < (unsigned)&start2)
    ? (void *)&start2 : (void *)&start1,
#endif

#ifdef USE_SIMPLE_GC
  GC_set_stack_base(mzscheme_stack_start);
#endif
#if defined(_IBMR2) && !defined(USE_SIMPLE_GC)
  if ((unsigned long)&no_rep > (unsigned long)0x2ff23000)
    scheme_stackbottom = 0x2ff80000;
  else
    scheme_stackbottom = 0x2ff23000;
  DynaPrintf("scheme_stackbottom %4x: \n",scheme_stackbottom);
#endif
  GC_allocate_ml = (HANDLE)NULL,
   f(!global_env) {
    global_env = scheme_basic_env();
    /* scheme_init_dynaplay(global_env); */
  }
}

/***********************************************
   type : char *      -> 1
          fixnum      -> 2
          others      -> 0

   return value   char   -> pointer
                  fixnum -> pointer
        if function returns #f, return value 0
 ***********************************************/
unsigned long DynaEvalString(const char *eval_body,int *type)
{
    Scheme_Object *ret;
    unsigned long val;
    jmp_buf savebuf;
    LOCK(EVAL);

    ret = (Scheme_Object *)NULL;
    *type = 0;

    memcpy(&savebuf, &scheme_error_buf, sizeof(jmp_buf));

    if (!scheme_setjmp(scheme_error_buf)) {
        ret = scheme_eval_string(eval_body,global_env);
    }
    memcpy(&scheme_error_buf, &savebuf, sizeof(jmp_buf)),

    if(ret) {
        if(SCHEME_STRINGP(ret)) {
            val = (unsigned long)SCHEME_STR_VAL(ret),
            *type = 1,
        } else if (SCHEME_INTP(ret)) {
            val = (unsigned long)SCHEME_INT_VAL(ret),
            *type = 2;
        }
        else {
            *type = 0;
            if(ret == scheme_false)
                val = 0;
            else
                val = 1;
        }
    } else {
        *type = 0;
        val = 0;
    }
    UNLOCK(EVAL),
    return(val);

}

int LoadNewScript(char *file_name)
{
  scheme_load(file_name),
```

```
  return 0;
}
```

Dynamap EXE

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <malloc.h>

#define SEPARATORS " "
#define END_OF_LINE_SEPARATOR " \t\n"
#define DEF_FILE_PREAMBLE ";%s\nNAME \"%s.exe\"\nEXPORTS\n"
#define DEF_FILE_PREAMBLE_DYNAMOD ";%s\nLIBRARY \"%s.dll\"\nEXPORTS\n"
#define F "f"
#define BUFSIZE 2048
#define PREFERRED "Preferred"
#define LOAD "load"
#define IS "is"
#define aDDRESS "address"
#define ADDRESS "Address"
#define PUBLICS "Publics"
#define BY  "by"
#define VALUE "Value"
#define COLON ':'
#define COMPARE_ADDRESS 14
#define COMPARE_SYMBOL 26
#define DYNA "?DYNA"
#define DFLAG "/D"
#define AFLAG "/A"
#define BASE 16
#define TRUE 1
#define DATA "DATA"
#define DATA_NULL ""
#define GLOBAL_COMMENT "; Global variables start here (I hope)"
#define DEF "def"

/*
 * This program takes a program map file (.map) and converts it
 * into a module definition file (.def) that will be used to
 * create an import library (.lib) that will satisfy any and all
 * external references in the dynamodule (.dll) that exist solely in the
 * original program (.exe).  This results in a dynamodule that is
 * the smallest possible size.
 */


int find_string(char *src, long *base_address)

{
      static char *stop_buf[] = {ADDRESS, PUBLICS, BY, VALUE, "\0"};
      static char *address_buf[] = {PREFERRED, LOAD, aDDRESS, IS, "\0"};
      char **continue_search;
      char *token;

      static base_flag = 1;

      // Return if not end of mapfile preamble or base address
      if ((token = strtok(src,SEPARATORS)) == NULL) {
            perror("strtok");
            return -1;
      }
      if (base_flag) {
            continue_search = address_buf;
      } else {
            continue_search = stop_buf;
```

```
        }
        if (strcmp(token,*continue_search++)) {
              return 0;
        }
        // Make sure
        while (**continue_search) {
              if ((token = strtok(NULL,SEPARATORS)) == NULL) {
                    perror("strtok");
                    return -1;
              }
              if (strcmp(token,*continue_search++)) {
                    return 0;
              }
        }
        // Get the base address
        if (base_flag) {
              if ((token = strtok(NULL,SEPARATORS)) == NULL) {
                    perror("strtok");
                    return -1;
              }
              if ((*base_address = strtol(token, (char **)NULL, BASE)) ==
0) {
                    perror("strtol");
                    return -1;
              }
              base_flag--;
              continue_search = stop_buf;
              return 0;
        }
        return 1;
}


int get_symbol(char *src, char **symbol, int *skip, int *fileflag, int
dynaflag, char **address)
{

        char *filename;
        static char tbuf[BUFSIZE];

        // Copy string to temporary buffer since strtok is destructive and
we still need the entire string
        if (strcpy(tbuf,src) == NULL) {
              perror("strcpy");
              return -1;
        }
        // Skip section information
        if ((*symbol = strtok(tbuf,SEPARATORS)) == NULL) {
              perror("strtok");
              return -1;
        }
        // Read symbol name
        if ((*symbol = strtok(NULL,SEPARATORS)) == NULL) {
              perror("strtok");
              return -1;
        }
        // Get address
        if ((*address = strtok(NULL,SEPARATORS)) == NULL) {
              perror("strtok");
              return -1;
        }
```

```
      // Get function declarator
      if ((filename = strtok(NULL,SEPARATORS)) == NULL) {
            perror("strtok");
            return -1;
      }
      // Check to see if token is filename or not
      if (strcmp(filename,F) == 0) {
      // Differentiate globals from functions
            (*fileflag)++;
      // Read filename
            if ((filename = strtok(NULL,SEPARATORS)) == NULL) {
                  perror("strtok");
                  return -1;
            }
      }
      /*
       * Eliminate entries that are dynamically linked.
       * We only want references that are statically linked in the .exe
because
       * Microsoft won't let me search this import lib last without
       * putting all the default libraries on the link command line.
       * Makes for a smaller import lib anyway.
       * Also skip if global variable and dynamod .def file is selected.
       */
      if (strchr(filename,COLON) != NULL || (dynaflag && (!*fileflag)))
{
            (*skip)++;
            if (*fileflag) {
                  (*fileflag)--;
            }
      }
      return 0;
}

int skip_map_file_preamble(FILE *fp, char *buf, long *base_address)
{
      int again = 1;

      /* read until 'Address Publics by Value' */
      while (again) {
            if(fgets(buf, BUFSIZE, fp) == NULL) {
                  perror("fgets");
                  return(-1);
            }
            switch (find_string(buf, base_address)){
            case 0: break;
            case 1:     again--;break;
            default: return(-1);
            }
      }
      // Skip to first symbol entry
    if(fgets(buf, BUFSIZE, fp) == NULL)  {
            perror("fgets");
            return(-1);
    }

      return 0;
}

char *get_program_name(FILE *map_file_name, char *buf)
{
.
```

```
    char *tok_ptr;

    // Read first line of mapfile to get program name
    if(fgets(buf, BUFSIZE, map_file_name) == NULL) {
            perror("fgets");
            return NULL;
    }
    // Strip end of line-
    if ((tok_ptr = strtok(buf,END_OF_LINE_SEPARATOR)) == NULL) {
        perror("strtok");
        return NULL;
    }
    return tok_ptr;
}

int write_def_file_preamble(FILE *fp, char *program_name, int dynaflag,
char *def_file_path)
{
    // Set up the .def file
    if (fprintf(fp,
dynaflag?DEF_FILE_PREAMBLE_DYNAMOD:DEF_FILE_PREAMBLE, def_file_path,
program_name) < 0) {
        perror("fprintf");
        return -1;
    }
}

int write_def_file(FILE *fp, char *symbol, int *fileflag, int dynaflag,
long offset)
{
    static ordinal = 1;
    static first_global = 1;

    // Write the symbols out, removing any leading '_', to the .def
file
    if (!dynaflag) {
        if (!(*fileflag) && (first_global)) {
            fprintf(fp,"%s\n", GLOBAL_COMMENT);
            first_global--;
        }
        if (fprintf(fp,"%s @%d NONAME %s\n",symbol[0] ==
'_'?++symbol:symbol, ordinal++, (*fileflag)?DATA_NULL:DATA) < 0) {
            perror("fprintf");
            return -1;
        }
    // No global variables in the dynamod file
    } else {
        if (*fileflag) {
            if (fprintf(fp,"%s @%d NONAME 0x%x\n",symbol[0] ==
'_'?++symbol:symbol, ordinal++, offset) < 0) {
                perror("fprintf");
                return -1;
            }
        }
    }

    if (*fileflag) {
        (*fileflag)--;
    }
    return 0;
}
```

```
int read_map_file(FILE *fp, char **symbol, int *skip, int *fileflag, int
dynaflag, char **address)
{
      char *bufp;

      static next_flag = 0;
      static again_flag = 1;
      static char buf[BUFSIZE],tempbuf[BUFSIZE];

      // Determine which buffer to use
      if (next_flag) {
            bufp = tempbuf;
            next_flag--;
      } else {
            bufp = buf;
            next_flag++;
      }
      // Get next line
      if(fgets(bufp, BUFSIZE, fp) == NULL) {
          perror("fgets");
            return -1;
      }
      // Get next line if necessary
      if (again_flag) {
            again_flag--;
            if (next_flag) {
                  bufp = tempbuf;
                  next_flag--;
            } else {
                  bufp = buf;
                  next_flag++;
            }
            if(fgets(bufp, BUFSIZE, fp) == NULL) {
             perror("fgets");
                  return -1;
            }
      }
      // Compare address of symbols
      if(strncmp(tempbuf, buf, COMPARE_ADDRESS) == 0) {
            // If same choose the one the linker won't complain about!
            if(strncmp(tempbuf, buf, COMPARE_SYMBOL) > 0) {
                  next_flag = 1;
            } else {
                  next_flag = 0;
            }
            //  Need two fgets instead of one next time
            again_flag++;
      }
      // Quit if done
      if (bufp[0] != ' ') {
            return 0;
      }

      if(get_symbol(next_flag?tempbuf:buf, symbol, skip, fileflag,
dynaflag, address)) {
            printf("Cannot get_symbol\n");
            return -1;
      }

      //next_flag ? next_flag-- : next_flag++;
```

```
        return 1;
}

/* Arguments to main:
 * argv[1] = /A or /D depending on context
 * argv[2] = name of map file (.map)
 * argv[3] = name of module definition file (.def)
 *
 * Returns:
 *  0 on success
 * -1 on error
 */

void main(int argc, char **argv)
{
        char *symbol, *name_buf, *address, buf[BUFSIZE],
rel_path_buf[BUFSIZE],
                        abs_path_buf[BUFSIZE];
        FILE *map_file_name, *def_file_name;
        long base_address, offset;

        int skip = 0;
        int fileflag = 0;
        int dynaflag = 0;

        // Parse args for correct flag and arg count.
        if (argc != 4) {
                        printf("Incorrect Argument Count\n");
                        exit(-1);
        }
        if (strcmp(argv[1],DFLAG) == 0) {
                dynaflag++;
        } else {
                if (strcmp(argv[1],AFLAG) != 0) {
                        printf("Incorrect Arguement.  Argument 1 must be
either /D or /A\n");
                        exit(-1);
                }
        }
        // Open mapfile
        if((map_file_name = fopen(argv[2],"r")) == NULL) {
                perror("fopen");
                printf("Cannot open %s \n",argv[2]);
                exit(-1);
        }
        // Create module definition file
        if((def_file_name = fopen(argv[3],"w")) == NULL) {
                perror("fopen");
                printf("Cannot create %s \n",argv[3]);
                exit(-1);
        }
        // Get program name
        if ((name_buf = get_program_name(map_file_name, buf)) == NULL) {
                exit(-1);
        }
        // Get location of application .def file
        if (strcpy(rel_path_buf, argv[2]) == NULL) {
                perror("strcpy");
                exit(-1);
        }
```

```
      if (strcpy(rel_path_buf + strlen(rel_path_buf) - sizeof(DEF)+1,
DEF) == NULL) {
            perror("strcpy");
            exit(-1);
      }
      // Convert relatvie to absolute path
      if (_fullpath(abs_path_buf,rel_path_buf,BUFSIZE) == NULL) {
            perror("_fullpath");
            exit(-1);
      }
      // Write module definition file preamble
      if(write_def_file_preamble(def_file_name, name_buf, dynaflag,
abs_path_buf) == -1) {
            printf("Cannot write module definition file preamble %s
\n",map_file_name);
            exit(-1);
      }
      // Skip mapfile preamble
      if(skip_map_file_preamble(map_file_name, buf, &base_address) == -
1) {
            printf("Cannot skip mapfile preamble %s \n",map_file_name);
            exit(-1);
      }
      // Read and write until done or error
      while (TRUE) {
            switch (read_map_file(map_file_name, &symbol, &skip,
&fileflag, dynaflag, &address)) {
            // Not Done
            case 1: break;
            // Done
            case 0: exit(0);
            // Error
            default: exit(-1);
            }
            // Skip symbol if not statically linked in the .exe
            if (skip) {
                  skip--;
            } else {
                  // If dynamod .def file, calculate funtion offset
                  if (dynaflag) {
                        if ((offset = strtol(address, (char **)NULL,
BASE)) == 0) {
                              perror("strtol");
                              exit(-1);
                        }
                        offset -= base_address;
                  }
                  if(write_def_file(def_file_name, symbol, &fileflag,
dynaflag, offset) == -1) {
                        exit(-1);
                  }
            }
      }
}
```

```
========================================================================
============
       MICROSOFT FOUNDATION CLASS LIBRARY : dynaplay
========================================================================
============


AppWizard has created this dynaplay DLL for you.  This DLL n
ot only
demonstrates the basics of using the Microsoft Foundation cl
asses but
is also a starting point for writing your DLL.

This file contains a summary of what you will find in each o
f the files that
make up your dynaplay DLL.

dynaplay.cpp
    This is the main DLL source file that contains the defin
ition of
        DllMain().


dynaplay.rc
    This is a listing of all of the Microsoft Windows resour
ces that the
    program uses.  It includes the icons, bitmaps, and curso
rs that are stored
    in the RES subdirectory.  This file can be directly edit
ed in Microsoft
        Developer Studio.

res\dynaplay.rc2
    This file contains resources that are not edited by Micr
osoft
        Developer Studio.  You should place all resources no
t
        editable by the resource editor in this file.

dynaplay.def
    This file contains information about the DLL that must b
e
    provided to run with Microsoft Windows.  It defines para
meters
    such as the name and description of the DLL.  It also ex
ports
```

```
        functions from the DLL.

dynaplay.clw
    This file contains information used by ClassWizard to ed
it existing
    classes or add new classes.  ClassWizard also uses this
file to store
    information needed to create and edit message maps and d
ialog data
    maps and to create prototype member functions.

/////////////////////////////////////////////////////////////
/////////////////
Other standard files:

StdAfx.h, StdAfx.cpp
    These files are used to build a precompiled header (PCH)
 file
    named dynaplay.pch and a precompiled types file named St
dAfx.obj.

Resource.h
    This is the standard header file, which defines new reso
urce IDs.
    Microsoft Developer Studio reads and updates this file.

/////////////////////////////////////////////////////////////
/////////////////
Other notes:

AppWizard uses "TODO:" to indicate parts of the source code
you
should add to or customize.

/////////////////////////////////////////////////////////////
/////////////////
```

Dynamod.bat

```
echo off
if "%OS%"=="Windows_NT" goto :NT
if not "%OS%"=="" goto :Error
command /e:4096 /c Dynabat2 %1 %2 %3 %4 %5 %6 %7 %8 %9
exit
:NT
Dynabat2 %1 %2 %3 %4 %5 %6 %7 %8 %9
exit
:Error
echo Dynamize: Environment variable "OS" must be either "Windows_NT"
echo           when running on NT or blank for Windows95.
```

Dynabat2.bat

```
echo off
rem ---------------------
rem Check the Arg count
rem ---------------------
if '%6' == '' goto :Args
if not '%7' == '' goto :Args
rem ---------------------
rem Check the Configuration
rem ---------------------
set CFG=Unknown
if %6 == DynaDebug set CFG="%2 - Win32 DynaDebug"
if %6 == DynaRelease set CFG="%2 - Win32 DynaRelease"
if %CFG% == Unknown goto :Config
rem ---------------------
rem Check for existence of application .def file
rem ---------------------
if not exist %4\%6.def goto :Def_error
rem ---------------------
rem Build the module definition file
rem ---------------------
if exist %1\%2.def erase %1\%2.def
Dynamap /D %1/%2.map %1/%2.def
if not exist %1\%2.def goto :Map_error
rem ---------------------
rem Generate the control file
rem ---------------------
Dynagen %4/%6.def %1/%2.def
if not errorlevel 0 goto :Gen_error
rem ---------------------
rem Done!
rem ---------------------
:Done
exit
rem ---------------------
rem Handle the errors
rem ---------------------
rem ---------------------
rem Remove any intermediate files
rem ---------------------
:Error
if exist %1\dynaplay.def erase %1\dynaplay.def
touch %1\dynaplay.def
if exist %1\%2.dll erase %1\%2.dll
exit
:Args
rem ---------------------
rem Complain about the argument count
rem ---------------------
echo Error: Dynamize: Wrong number of arguments.
echo Custom build command for project %3 must be the following:
echo Dynamod "$(OutDir)" "$(InputName)" "$(Wkspname)" "$(WkspDir)" "$(IntDir)"
DynaDebug
echo Or
echo Dynamod "$(OutDir)" "$(InputName)" "$(Wkspname)" "$(WkspDir)" "$(IntDir)"
DynaRelease
echo Depending on which build configuration you are running.
goto :Error
```

```
:Config
rem ---------------------
rem Complain about the configuration parameter
rem ---------------------
echo Error: Dynamize: Unknown Configuration.
echo Custom build parameter 5 for project %3 is %6.
echo Custom build parameter 5 must be either DynaDebug or DynaRelease.
echo Check your custom build settings for the current configuration.
goto :Error
rem ---------------------
rem Complain about .def file
rem ---------------------
:Def_error
echo Error: Dynamod: Could not find file %4\%6.def
echo Please rebuild application
goto :Error
rem ---------------------
rem Complain about Dynamap
rem ---------------------
:Map_error
echo Error: Dynamod: Could not generate module definition file
echo Make sure that file Dynamap.exe is in your search path
goto :Error
rem ---------------------
rem Complain about Dynagen
rem ---------------------
:Gen_error
echo Error: Dynamod: Could not generate virtual filesystem
echo Make sure that file Dynagen.exe is in your search path
goto :Error
```

Dynabat.bat

```
echo off
rem --------------------
rem Check the Arg count
rem ---------------------
if '%7' == '' goto :Args
if not '%8'== '' goto :Args
rem ---------------------
rem Check the Configuration
rem ---------------------
set CFG=Unknown
set DYNALIB=%5\%7.lib
if %7 == DynaDebug set CFG="%2 - Win32 DynaDebug"
if %7 == DynaRelease set CFG="%2 - Win32 DynaRelease"
if %CFG% == Unknown goto :Config
rem ---------------------
rem Remove the application
rem ---------------------
if exist %1\%2.exe erase %1\%2.exe
if exist %1\%2.exe goto :App_error
rem ---------------------
rem Build the module definition file
rem ---------------------
if exist %5\%7.def erase %5\%7.def
Dynamap /A %1/%2.map %5/%7.def
if not exist %5\%7.def goto :Map_error
rem ---------------------
rem Generate the .lib
rem --------------------
if exist %5\%7.lib erase %5\%7.lib
link /lib /nologo /def:%5\%7.def /out:%5\dynaplay.lib > %1\dynagarbage.can
copy %5\dynaplay.lib %DYNALIB% > %1\dynagarbage.can
if not exist %DYNALIB% goto :Lib_error
rem ---------------------
rem Generate the .dbj's
rem --------------------
if exist %1\dynatab.dbj erase %1\dynatab.dbj
Dynaobj %1 /A %5/%7.def
if not exist %1\dynatab.dbj  goto :Dbj_error
rem ---------------------
rem Move the .obj's to .obd's
rem Move the .dbj's to .obj's
rem --------------------
if exist %1\*.obd erase %1\*.obd
rename %1\*.obj *.obd
rename %1\*.dbj *.obj
copy %1\dynatab.obj . > %1\dynagarbage.can
if exist %1\*.dbj goto :Rename_error
rem ---------------------
rem Check for existence of makefile
rem ---------------------
if not exist %3.mak goto :Export_error
rem ---------------------
rem Relink the application with the export file
rem ---------------------
nmake /nologo /s /f %3.mak %4 CFG=%CFG%
if errorlevel 1 goto :Nmake_error
rem ---------------------
```

to link the library and create Dynaplay.lib Dynaplay.exe

```
rem Rerename the .dbj's and .obj's
rem ---------------------
if exist %1\*.dbj erase /q %1\*.dbj
rename %1\*.obj *.dbj
rename %1\*.obd *.obj
if exist %1\*.obd goto :Rename_error
rem ---------------------
rem Create the Virtual File System
rem ---------------------
Dynagen %5/%7.def
if not errorlevel 0 goto :Vfs_error
rem ---------------------
rem Remove the extraneous files
rem ---------------------
if not exist %5\dynaplay.exp goto :Exp_error
if exist %5\dynaplay.exp erase %5\dynaplay.exp
if exist %5\dynatab.obj erase %5\dynatab.obj
if exist %1\%2.map erase %1\%2.map
if exist %1\dynagarbage.can erase %1\dynagarbage.can
rem ---------------------
rem Done!
rem ---------------------
exit
rem ---------------------
rem Handle the errors
rem ---------------------
rem ---------------------
rem Remove any intermediate files
rem ---------------------
:Error
if exist %4 erase %4
if exist %5\%7.def erase %5\%7.def
if exist %DYNALIB% erase %DYNALIB%
if exist %5\dynaplay.exp erase %5\dynaplay.exp
if exist %5\dynaplay.lib erase %5\dynaplay.lib
if exist %5\dynatab.obj erase %5\dynatab.obj
if exist %1\dynagarbage.can erase %1\dynagarbage.can
exit
:Args
rem ---------------------
rem Complain about the argument count
rem ---------------------
echo Error: Dynamize: Wrong number of arguments.
echo Custom build command for project %3 must be the following:
echo Dynamize "$(OutDir)" "$(InputName) "$(WkspName) "$(TargetPath)"
"$(WkspDir)" "($IntDir)" DynaDebug
echo Or
echo Dynamize "$(OutDir)" "$(InputName) "$(WkspName) "$(TargetPath)"
"$(WkspDir)" "($IntDir)" DynaRelease
echo Depending on which build configuration you are running.
goto :Error
:Config
rem ---------------------
rem Complain about the configuration parameter
rem ---------------------
echo Error: Dynamize: Unknown Configuration.
echo Custom build parameter 6 for project %3 is %7.
echo Custom build parameter 6 must be either DynaDebug or DynaRelease.
echo Check your custom build settings for the current configuration.
goto :Error
rem ---------------------
rem Complain about application
rem ---------------------
:App_error
echo Error: Dynamize: Could not erase file .\%2.exe
goto :Error
rem ---------------------
rem Complain about Dynamap
rem ---------------------
:Map_error
echo Error: Dynamize: Could not generate module definition file
echo Make sure that file Dynamap.exe is in your search path
goto :Error
rem ---------------------
rem Complain about lib.exe
rem ---------------------
:Lib_error
echo Error: Dynamize: Could not sucessfully execute lib.exe
echo Possible causes include :
echo                          Could not find lib.exe
echo                          Wrong or corrupt version of lib.exe
echo                          Missing or corrupt file %1\%2.def
echo                          Not running on an Intel cpu based machine
goto :Error
rem ---------------------
rem Complain about Dynaobj.exe
rem ---------------------
:Dbj_error
echo Error: Dynamize: Could not generate dynatab.dbj
echo Make sure Dynaobj.exe is in your search path
if exist %1\dynatab.dbj erase %1\dynatab.dbj
goto :Error
rem ---------------------
rem Complain about renaming files
rem ---------------------
:Rename_error
echo Error: Dynamize: Could not rename files
echo Make sure .dbj and .obj files exist
goto :Reset_files
rem ---------------------
rem Complain about existence of makefile
rem ---------------------
:Export_error
echo Error: Dynamize: No makefile exists for this project.
echo                  Please export makefile and rebuild.
goto :Reset_files
rem ---------------------
rem Complain about nmake
rem ---------------------
:Nmake_error
echo Error: Dynamize: Could not run nmake -f %2.mak %4 %CFG%
echo Seek professional help
goto :Reset_files
rem ---------------------
rem Remove the export file
rem ---------------------
:Exp_error
echo Error: Dynamize: Could not erase file %5\%2.exp
goto :Error
rem ---------------------
rem Rerename the .obj's and .dbj's
rem ---------------------
:Vfs_error
echo Error: Dynamize: Could not create Virtual File System
echo        Go bother Takashi
goto :Error
rem ---------------------
rem Rerename the .obj's and .dbj's
rem ---------------------
:Reset_files
if exist %1\*.dbj erase /q %1\*.dbj
rename %1\*.obj *.dbj
rename %1\*.obd *.obj
goto :Error
```

Dynamize.bat

```
echo off
if "%OS%"=="Windows_NT" goto :NT
if not "%OS%"=="" goto :Error
command /e:4096 /c Dynabat %1 %2 %3 %4 %5 %6 %7 %8 %9
exit
:NT
Dynabat %1 %2 %3 %4 %5 %6 %7 %8 %9
exit
:Error
echo Dynamize: Environment variable "OS" must be either "Windows_NT"
echo           when running on NT or blank for Windows95.
```

DynaPlay SDK README FILE
DynaPlay SDK Version 0.20

Configuration instructions for using DynaPlay with Microsoft Developer's Studio.

<u>**IMPORTANT NOTE!**</u>:

Currently the DynaPlay SDK is only configured to support Microsoft Developer's Studio versions 4.x with the exception of version 4.0 under NT. Release 5.0 is **NOT** supported at this time. Furthermore, the DynaPlay SDK will **NOT** work with version 5.0 at this time. Furthermore, it's Microsoft's fault, not mine, at this time.

ALSO NOTE:

An example for the following instructions exists in the DynaPlay SDK. Refer to the Hangman32 workspace provided in

c:\Program Files\DynaPlay\Examples\Hangman32

(or wherever you installed the SDK) as an example of how to set up a DynaPlay workspace. Only steps 1 and 9 need be performed in order to build and execute this example.

In order to use the DynaPlay SDK with Developer's Studio versions 4.x, the following steps must be taken after the DynaPlay SDK has been successfully installed:

1. Make sure the DynaPlay executables and libraries are in the Developer's Studio search path. Add these directories (typically, c:\Program Files\DynaPlay\bin and c:\Program Files\DynaPlay\lib) to your executable and library search paths by calling up the Tools Options window and clicking on the Directories option. Add

c:\Program Files\DynaPlay\bin

(or wherever you installed the SDK) under executable files and

c:\Program Files\DynaPlay\lib

(or wherever you installed the SDK) under library files.

Note: Check to see if the file

dynalib.dll

was installed in your Windows system directory (usually \Windows\System for Windows95 and \Winnt\System32 for NT). If you do not have write access to this directory, dynalib.dll will be installed in

c:\Program Files\DynaPlay\bin

(or wherever you installed the SDK). This will allow you to build your applications from within Developer's Studio but if you wish to execute them you must make sure that dynalib.dll is placed in a directory that is included in your **PATH** environment variable.

2. Create the DynaPlay build configurations for your existing project. These configurations must be named

DynaDebug

and

DynaRelease

Create these by invoking the Build Configurations menu option and selecting Add. Use the configuration settings for your Debug configuration in creating the DynaDebug configuration and similarly use the configuration settings for your Release configuration in creating the DynaRelease configuration.

3. After the new configurations have been created, call up the Build Settings window and click on the DynaDebug configuration. Hold the Ctrl key down while clicking on the DynaRelease configuration so that both configurations are selected. Now any modifications you make to the build settings will apply to both configurations.

4. Under the General category for Build Settings, make sure that the option

Use MFC in a shared DLL

is selected. This is necessary for all MFC based DynaPlay applications and has no effect if your application is not MFC based.

5. Under the C/C++ category select Optimizations. For the In-line function expansion setting, choose

Disable *

This is not absolutely necessary but is highly recommended until you have gained enough experience with DynaPlay to know how much trouble you can get into by not selecting this option.

6. Under the Link category select General. For the option Object/library modules, add the following:

dynatab.obj dynaplay.exp dynalib.lib

Also, make sure the

Generate mapfile

option is selected and that the

Link Incrementally

option is turned off, otherwise you will get an annoying warning message every time you build since incremental linking is incompatible with map generation.

7. Call up the Custom Build category and under Build command(s), add the following line:

```
Dynamize "$(OutDir)" "$(InputName)" "$(WkspName)" "$(TargetPath)" "$(WkspDir)" "$(IntDir)" DynaReleas
```

You might try cutting and pasting the line above to make sure you get it right. For the Output file(s) section, add the following:

```
$(WkspDir)\DynaRelease.def
```

NOTE:
If you can see the difference between the workspace name and the project name, you will have to use the project name instead of **"$(InputName)"**. **"$(InputName)"** represents the workspace name. The project name is shown in the project workspace window.
For example, the workspace name is **"Test"** and the project name is **"test"** the following:

```
Dynamize "$(OutDir)" "test" "$(WkspName)" "$(TargetPath)" "$(WkspDir)" "$(IntDir)" DynaRelease
```

8. Now click on the DynaDebug configuration so that only it is selected. Change the Custom Build information for this configuration so that all references to DynaRelease are changed to DynaDebug. The resulting lines should look like:

```
Dynamize "$(OutDir)" "$(InputName)" "$(WkspName)" "$(TargetPath)" "$(WkspDir)" "$(IntDir)" DynaDebug
```

to the Build command(s) section and

```
$(WkspDir)\DynaDebug.def
```

to the Output file(s) section. Finally, in the Link Project Options section, add the following option:

```
/opt:noref
```

Make sure that you scroll down to the bottom of the Project Options before entering the above option since Developer's Studio is likely to misinterpret how this option is to be parsed otherwise.

NOTE:
If you can see the difference between the workspace name and the project name, you will have to use the project name instead of **"$(InputName)"**. **"$(InputName)"** represents the workspace name. The project name is shown in the project workspace window.
For example, the workspace name is **"Test"** and the project name is **"test"** the following:

Dynamize "$(OutDir)" "test" "$(WkspName)" "$(TargetPath)" "$(WkspDir)" "$(IntDir)" DynaDebug

9. Build and execute the application in both the DynaDebug and DynaRelease configuration to make sure that everything is configured correctly. If so, you have now successfully built an application that has been "Dynamized" and is now ready to accept DynaPlay modules. Of note is the fact that you have accomplished this without making any modifications to the existing source code for the application. You are now ready to create the modules that DynaPlay uses to modify your existing application. These modules can be applied both at application startup and dynamically during the runtime of the application to modify the behavior of the application in any manner the programmer chooses. Not only can these modifications be performed dynamically (hence, the name DynaPlay) but these modifications require no forethought, that is, they can be made to the application after it has been created and distributed without prior planning as to the nature of these changes. The next section will discuss how to create the modules that DynaPlay ready applications use.

NOTE:
Do not change **Intermediate files** and **Output files** under the General category for Build Settings.

Creating modules for use with DynaPlay ready applications

At present, the responsibility for maintaining the source code used in the **Dynamodules**, rests with the user. This can be accomplished in a number of different ways, depending upon the nature of the application being developed. The Hangman example provided with the SDK shows separate directories, **dyna_include** and **dyna_c** for holding the source code used in the **Dynamodule**. This approach allows complete freedom to modify the application in any way, including header file modifications, without affecting the source code for the original application. This could be accomplished in other ways, such as using Visual SourceSafe to keep track of different versions of the code or the user may simply decide to modify the original code itself. At present, the choice of which method to use is up to the user. Keep in mind the following requirements when making your decision:

1. The source code contained in the **Dynamodule** consists of only the changes you wish to make to the original application. Only those functions that are modified are included in the **Dynamodule**.

2. The filenames of the **Dynamodule** source code must have the same names as the filenames in the original application. For example, if you modify function **x()** in file **a.c** and function **y()** in file **b.c**, your **Dynamodule** will consist of code from two files, **a.c** and **b.c**. **a.c** will contain only function **x()** and **b.c** will contain only function **y()**. You cannot combine both functions into one file and call it, for example, **c.c**(or **a.c** or **b.c** for that matter).

Once the decision has been made as to how the **Dynamodule** source code will be maintained, the procedure for creating a **Dynamodule** project in Developer Studio is a follows:

1. Open the workspace that includes the original application.

2 Select the menu option **Insert Project** and *choose* to create either a **Dynamic-Link Library** or an **MFC AppWizard(dll)** depending on whether or not your application is an **MFC** application. Create this dll as a Top level project. You may name it anything you wish.

Note: If using **MFC AppWizard(dll)** to create the **Dynamodule**, you must select the **MFC extension dll (using shared MFC DLL)** option during the creation process.

3. If you have created an **MFC Dynamodule**, you must replace the **"resource.h"** file created by the **AppWizard** for the **Dynamodule** with the **"resource.h"** file used by the application. Use Windows Explorer to copy **"resource.h"** from the project directory of the application to the project directory of the **Dynamodule**, replying yes when asked if you wish to replace the existing file.

4. Select the menu option **Build Subprojects**. For the **Dynamodule** project you just created, include the original application as a subproject. This will insure that if the original application changes in any way that it will be rebuilt before the **Dynamodule** is built.

5. Select the **Build Configurations** menu option and, as you did with the original application, create two new configurations, **DynaDebug** and **DynaRelease.**

6. Include two files that were created when the original application was Dynamized, **DynaDebug.lib** and **DynaRelease.lib**. They should reside in the root directory of this project's workspace.

7. Transfer any pertinent **Build Settings** from the original application to the **Dynamodule**. With both the **DynaDebug** and **DynaRelease** configurations selected, make sure the following options are set:

   a. Under the **General** category, make sure that **Use MFC in a Shared Dll** is selected, if this is an MFC application. This is necessary even though the **Dynamodule** was created as a non-MFC dll.

   b. Under the **C/C++ optimizations** category, make sure that **In-line function expansion** is set to **Disable ***.

   c. Under the **Link General** category, make sure that **Generate mapfile** is selected.

8. Select just the **DynaRelease** build setting and set the following options:

   a. Under the **Link Project Options** add the following:

   /opt:noref

   just as you did for the **DynaRelease** configuration of the original application.

   b. Under the **Custom Build** category, add the following **Build command**:

   ```
   Dynamod "$(OutDir)" "$(InputName)" "$(WkspName)" "$(WkspDir)" "$(IntDir)" DynaRelease
   ```

   As with the original application, you may find cutting and pasting to be useful here.

c. For the **Output file(s)** enter the following:

$(OutDir)\$(InputName).def

9. Select just the **DynaDebug** build setting and set the following options:

   a. Under the **Custom Build** category, add the following **Build command**:

   Dynamod "$(OutDir)" "$(InputName)" "$(WkspName)" "$(WkspDir)" "$(IntDir)" DynaDebug

   b. For the **Output file(s)** enter the following:

   $(OutDir)\$(InputName).def

10. Select the Build Settings for **DynaDebug.lib** under the **DynaRelease** configuration and set the following:

    a. Under the **General** category, make sure the **Exclude file from build** option is selected.

11. Select the Build Settings for **DynaRelease.lib** under the **DynaDebug** configuration and set the following:

    a. Under the **General** category, make sure the **Exclude file from build** option is selected.

12. Include the source files into the project that you wish to modify. Remember, include only those files that contain modifications to the original application or the dll will be unnecessarily large. Also remember that any files included in the project should contain only those functions that are modified for the same reason.

Note: MFC Dynamodules can still use **Class Wizard,** however, you must copy, exactly, the constructor function, the **AFX_DATA_MAP** and the **AFX_MSG_MAP** of the object and then manually add the object to the **Class Wizard** database. Refer to the example in the SDK for details.

13. New resources may be added to **MFC Dynamodules** as you would normally add them to the application. No additional code is required to manage their use. You may also modify existing resources by drag and dropping them from the application into the Dynamodule and then editing them. No additional code is required. Also, compound resources, such as dialog boxes that include icons, need not include those resources that will not be modified. This is a new feature, available only with **DynaPlay,** that eliminates the need for redundant resources in the **Dynamodule** resulting in a much smaller dll than is typically capable using traditional programming methods. Refer to the About box in the Hangman application for an example of how this can be used.

14. Build the application.

Note: The first time you try and build an **MFC Dynamodule,** you will be asked if you wish to overwrite the existing **"resource.h"** file. This is due to having copied the application's **"resource.h"** file in step 3. Answer yes to this question. You will not be asked again.

15. Select the **Execute** option. When prompted, enter the relative path name of the original application that was built for this **Dynamodule**. Remember that the **DynaDebug** version of the original application only works with **DynaDebug** versions of the **Dynamodules** and that the same holds true for the **DynaRelease** version of the original application with respect to the **DynaRelease** versions of any **Dynamodules** created.

The application should now execute with the code created in the **Dynamodule** substituted in place of the code created in the original application. Executing the Hangman32 example provided in the SDK demonstrates this.

```
; ############################################################
; #          Following commands You can Modify               #
; # * If you want to modify commands, you have to take       #
; #   out ';' at first line.                                 #
; # * The root of a terget path is the directory where       #
; #   is an application in user environmant.                 #
; # * /initscb is a control script that represents before   #
; #   before /initsc. /initsca means a control script that #
; #   represents after /initsc.                              #
; # * /initscb + /initsc + /initsca is a control script for#
; #   DynaModule. The control script saves XXX.dat           #
; # * /dependent:        is a dependent modules which is    #
; #   used by this DynaModule.                               #
; # * /eval: is the evaluating script in a DynaInstall.exe #
; # * Use '\' and CR to continure a line                     #
; ############################################################
;/scinitb: ControlScript_brefore_'initsc'
;/scinita: ControlScript_after_'initsc'
;/dependent:module_name1,module_name2,...
;/eval:ControlScript
;/vfs:Source_path|Target_path_in_VFS
;/vfsdata:Value|Target_path_in_VFS
/data:C:\Program Files\Net
Fighter\DynaModules\DynaModule1\DynaRelease\dynamodule1.dll|selfandheat\dynamo
dule1.dll
/data:C:\Program Files\Net
Fighter\DynaModules\DynaModule2\DynaRelease\dynamodule2.dll|selfandheat/dynamo
dule2.dll
/out:c:\temp\aaa.dyp
```

dyp - this is what user downloads

.dll + [illegible]

script - binary

binary means small file
+ can't read

user then runs dynainstall. ~~exe~~ exe to execute script, store data in VFS, store data in correct directories